Appunti per il corso di Ricerca Operativa

Capitolo 1

Introduzione

La Ricerca Operativa è una disciplina che si occupa di fornire strumenti di supporto per risolvere problemi di decisione complessi. Poco più avanti analizzeremo in dettaglio le componenti di un problema di decisione ma a grandi linee, è a tutti chiaro in cosa un tale problema consista: dato un insieme di scelte possibili, dobbiamo individuare la migliore rispetto a un qualche criterio. Riguardo la complessità va invece chiarito di quale genere è quella per cui ci possono essere di aiuto gli strumenti forniti dalla Ricerca Operativa. In alcuni casi può essere estremamente difficile individuare la scelta migliore anche quando l'insieme delle seclte possibili è estremamente limitato (si pensi, ad esempio, alla decisione se iscriversi o meno all'università, dove le scelte possibili sono soltanto due). *Non* è questo il genere di complessità per cui ci può aiutare la Ricerca Operativa. Questa viene utile quando l'insieme delle seclte possibili è di grandi dimensioni: in tal caso ci può fornire strumenti intelligenti per esplorare l'insieme alla ricerca della soluzione migliore.

Vediamo ora attraverso quali fasi si sviluppa il lavoro di un ricercatore operativo che debba risolvere un problema di decisione.

Individuazione delle componenti del problema di decisione

In un problema di decisione possiamo sempre riconsocere le seguenti componenti:

DATI : rappresentano tutte le informazioni note a priori (input del problema).

VARIABILI : sono le entità controllate dal decisore che ne può, appunto, variare il valore.

VINCOLI : limitano le possibili scelte del decisore (equivalentemente, i possibili valori delle variabili).

OBIETTIVO : coincide con il criterio fissato per confrontare le diverse possibili scelte del decisore.

Tabella 1.1:

	Farina	Acqua	Medicinali
TIPO I	10	10	30
TIPO II	30	20	10
TIPO III	20	40	5

Tabella 1.2:

TIPO I	14
TIPO II	5
TIPO III	4

Per comprendere meglio queste quattro componenti, consideriamo un paio di esempi di problemi di decisione. Il primo è banale e facilmente risolvibile, mentre il secondo, pur apparentemente semplice, non è facilmente risolvibile senza gli strumenti forniti dalla Ricerca Operativa.

Esempio 1 *Dovete uscire di casa e potete prendere con voi al massimo uno dei seguenti tre oggetti: un libro che vale 10 Euro, una macchina fotografica che vale 100 Euro e una borsa da 25 Euro. Dovete decidere quale oggetto portare con voi, tenuto conto che vi interessa prendere un oggetto di valore massimo. L'esempio è molto banale e non c'è bisogno di scomodare la Ricerca Operativa per capire che occorre prendere la macchina fotografica. Tuttavia in esso sono già presenti tutte le componenti tipiche di una decisione:*

DATI *: sono i valori dei tre oggetti.*

VARIABILI *: per ogni oggetto il decisore deve decidere se prenderlo oppure no.*

VINCOLI *: in questo caso è presente il vincolo che può essere preso al massimo un oggetto.*

OBIETTIVO *: il criterio di scelta è rappresentato dal valore e quindi l'obiettivo è quello di prendere l'oggetto di valore massimo.*

Esempio 2 *Supponiamo di dover preparare dei pacchi per inviare degli aiuti. È possibile realizzare tre diversi tipi di pacchi con diversi contenuti di sacchetti di farina, bottiglie d'acqua e medicinali. Più precisamente la Tabella 1.1 specifica i contenuti di ogni tipo di pacco. È stato inoltre assegnato un indice di utilità per un'unità di ogni tipo di pacco. Gli indici sono riportati nella Tabella 1.2. Infine è noto che la disponibilità di sacchetti di farina, bottiglie d'acqua e medicinali è limitata. Più precisamente il numero massimo disponibile di farina, acqua e medicinali è riportata nella Tabella 1.3. La domanda che ci si pone è la seguente: quanti pacchi di ciascun tipo occorre preparare se si vuole rendere massimo l'indice di utilità complessivo? Vediamo di individuare le quattro componenti del problema di decisione.*

Tabella 1.3:

farina	5100
acqua	8000
medicinali	1805

DATI *: sono i valori riportati nelle tre tabelle 1.1-1.3.*

VARIABILI *: per ogni tipo di pacco il decisore deve decidere quanti pacchi di quel tipo realizzare.*

VINCOLI *: in questo caso sono presenti i vincoli sulla disponibilità di farina, acqua e medicinali.*

OBIETTIVO *: il criterio di scelta è rappresentato dall'utilità complessiva dei pacchi, che si vuole massimizzare.*

Un problema di questo tipo viene chiamato problema con vincoli di risorse. In tali problemi vi sono sempre delle risorse (in questo caso farina, acqua e medicinali) che vengono in qualche modo utilizzate (qui per fare i pacchi) e delle quali si ha una disponibilità limitata.

*Notiamo che rispetto all'esempio iniziale è ora più difficile stabilire qual è la cosa giusta da fare. Potrei realizzare solo pacchi del tipo I. In tal caso ne potrei realizzare al massimo 60 per il limite di disponibilità sui medicinali. L'utilità complessiva risulterebbe pari a 60*14=840. Potrei realizzare solo pacchi del tipo II. In tal caso ne potrei realizzare al massimo 170 per il limite di disponibilità sui sacchetti di farina. L'utilità complessiva risulterebbe pari a 170*5=850. Infine, potrei realizzare solo pacchi del tipo III. In tal caso ne potrei realizzare al massimo 200 per il limite di disponibilità sulle bottiglie d'acqua. L'utilità complessiva risulterebbe pari a 200*4=800. Delle tre possibili soluzioni la migliore è la seconda. Ma queste tre scelte non coprono tutti i casi possibili. Infatti, potrei scegliere di fare un po' di pacchi di ciascun tipo. Quindi, a differenza dell'esempio iniziale non è per nulla immediato individuare la scelta migliore.*

Creazione di un modello matematico del problema

Un modo per risolvere problemi di decisione complessi è quello di riformularli come modelli di *Programmazione Matematica* e utilizzare quindi delle tecniche di risoluzione apposite per questi modelli. La riformulazione è in sostanza una *traduzione* del problema di decisione, di cui si ha una descrizione a parole, in un nuovo linguaggio che è il linguaggio matematico. Se il nostro problema ha n variabili, queste saranno rappresentate con opportuni simboli quali $x_1, \ldots, x_n$. Alcune di queste (diciamo le prime k) possono assumere valori reali, altre potrebbero essere vincolate ad assumere solo valori interi. L'obiettivo viene tradotto in una funzione matematica $f(x_1, \ldots, x_n)$ delle n variabili, detta appunto *funzione*

obiettivo. I vincoli vengono tradotti in disequazioni o equazioni di questo tipo:

$$g_i(x_1, \ldots, x_n) \leq (\text{o } \geq \text{ o } =)\ 0.$$

Quindi la generica forma di un problema di Programmazione Matematica è la seguente:

$$\begin{array}{ll} \max \text{ (o } \min) & f(x_1, \ldots, x_n) \\ & g_i(x_1, \ldots, x_n) \leq 0 \quad i \in I_1 \\ & g_i(x_1, \ldots, x_n) \geq 0 \quad i \in I_2 \\ & g_i(x_1, \ldots, x_n) = 0 \quad i \in I_3 \\ & x_1, \ldots, x_k \in R \\ & x_{k+1}, \ldots, x_n \in Z \end{array}$$

L'insieme dei punti che soddisfano tutti i vincoli viene chiamato *regione ammissibile* del problema e nel seguito verrà indicato con S. Abbiamo quindi:

$$\begin{array}{lll} S = \{ \quad (x_1, \ldots, x_n) : & g_i(x_1, \ldots, x_n) \leq 0, & \forall\ i \in I_1 \\ & g_i(x_1, \ldots, x_n) \geq 0, & \forall\ i \in I_2 \\ & g_i(x_1, \ldots, x_n) = 0, & \forall\ i \in I_3 \\ & x_1, \ldots, x_k \in R & \\ & x_{k+1}, \ldots, x_n \in Z\} & \end{array}$$

Risolvere il problema di Programmazione Matematica vuol dire determinare un punto $(x_1^*, \ldots, x_n^*) \in S$, che verrà detto *soluzione ottima* del problema, tale che

$$f(x_1^*, \ldots, x_n^*) \leq f(x_1, \ldots, x_n) \quad \forall\ (x_1, \ldots, x_n) \in S,$$

se il problema è di minimo, oppure

$$f(x_1^*, \ldots, x_n^*) \geq f(x_1, \ldots, x_n) \quad \forall\ (x_1, \ldots, x_n) \in S,$$

se il problema è di massimo.

La Programmazione Matematica comprende un grande numero di problemi. Tra questi una particolare rilevanza hanno i problemi di Programmazione Lineare (PL) e Programmazione Lineare Intera (PLI). Per entrambi questi problemi tutte le funzioni (quella obiettivo e quelle che definiscono i vincoli) sono funzioni lineari. La sola differenza tra PL e PLI è rappresentata dal fatto che nella PL sono presenti solo variabili reali, mentre nella PLI sono presenti solo variabili che possono assumere valori interi (si possono avere anche casi misti con alcune varaibili reali e altre intere ma qui non ce ne occuperemo visto che le tecnciche risolutive per questi sono analoghe a quelle per i problemi di PLI). Il generico

problema di PL avrà la seguente forma:

$$\begin{array}{ll} \max \text{ (o } \min) & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i \in I_1 \\ & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i \in I_2 \\ & \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i \in I_3 \\ & x_1, \ldots, x_n \in R \end{array}$$

mentre nei problemi di PLI la sola differenza sarà rappresentata dal vincolo sulle variabili che sarà

$$x_1, \ldots, x_n \in Z$$

I problemi lineari sono molto importanti perchè molti problemi reali hanno un modello di PL o PLI. Inoltre, per quanto riguarda la PL, la relativa semplicità di risoluzione di tali problemi (esistono per essa metodi di risoluzione molto efficienti) implica che anche metodi di risoluzione per problemi più complessi (tra cui anche quelli di PLI, come avremo modo di vedere) si basano sulla risoluzione multipla di problemi di PL.

Esempio 3 *Vediamo di ricavare il modello matematico per il problema degli aiuti umanitari. Indichiamo con:*

x_1 *il numero di pacchi di tipo I da realizzare.*

x_2 *il numero di pacchi di tipo II da realizzare.*

x_3 *il numero di pacchi di tipo III da realizzare.*

Queste sono le tre variabili del problema. Ora dobbiamo tradurre i vincoli e l'obiettivo in formule matematiche. Abbiamo un vincolo sulla disponibilità di sacchetti di farina. Come si può tradurre in linguaggio matematico? Un pacco di tipo I contiene 10 sacchetti di farina. Quindi x_1 pacchi di tipo I richiedono

$$10x_1$$

sacchetti di farina. Analogamente, un pacco di tipo II contiene 30 sacchetti di farina e quindi x_2 pacchi richiedono

$$30x_2$$

sacchetti di farina. Infine, un pacco di tipo III contiene 20 sacchetti di farina e quindi x_3 pacchi richiedono

$$20x_3$$

sacchetti di farina. La somma di questi tre valori restituisce il numero complessivo di sacchetti utilizzati. Più precisamente, il valore

$$10x_1 + 30x_2 + 20x_3$$

rappresenta il numero complessivo di sacchetti di farina utilizzati in corrispondenza dei valori x_1, x_2 e x_3 delle variabili. Noi sappiamo di non poter utilizzare più di 5100 sacchetti di farina e tale vincolo si traduce quindi nella seguente disequazione

$$10x_1 + 30x_2 + 20x_3 \leq 5100,$$

che è proprio la traduzione in linguaggio matematico del vincolo sulla disponibilità di sacchetti di farina. In modo completamente analogo si procede per tradurre i vincoli sulla disponibilità di bottiglie d'acqua

$$10x_1 + 20x_2 + 40x_3 \leq 8000,$$

e sulla disponibilità di medicinali

$$30x_1 + 10x_2 + 5x_3 \leq 1805.$$

Per essere precisi a questi tre vincoli ne dobbiamo aggiungere altri tre che non abbiamo specificato in precedenza perché banali: le quantità di pacchi di ciascun tipo non possono essere negative (per esempio, non ha senso parlare di -5 pacchi di tipo I). Mentre in una descrizione a voce del problema questo tipo di vincoli è del tutto scontato, da un punto di vista matematico non lo è e tali vincoli sono essenziali nella definizione del modello. In linguaggio matematico essi si esprimono semplicemente in questo modo:

$$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0.$$

L'insieme dei valori che si possono assegnare a x_1, x_2, x_3 senza violare i vincoli introdotti rappresenta l'insieme ammissibile S del problema. Ad esempio, $x_1 = 20, x_2 = 20, x_3 = 30$ è una soluzione ammissibile, mentre $x_1 = 50, x_2 = 60, x_3 = 40$ non lo è (viola il vincolo sulla disponibilità di medicinali) e non lo è neppure $x_1 = -2, x_2 = 20, x_3 = 40$ (viola il vincolo di non negatività del numero di pacchi di tipo I).
Resta da definire l'obiettivo del problema. Un pacco di tipo I ha utilità pari a 14, quindi x_1 pacchi hanno utilità pari a

$$14x_1.$$

In modo del tutto analogo si vede che x_2 pacchi di tipo II hanno utilità pari a

$$5x_2$$

e x_3 pacchi di tipo III hanno utilità pari a

$$4x_3.$$

Quindi, sommando le utilità di ciascun tipo di pacco si ottiene l'utilità complessiva pari a

$$14x_1 + 5x_2 + 4x_3.$$

Il nostro obiettivo è massimizzare tale valore.
Riassumendo, il modello matematico del nostro problema è il seguente:

$$
\begin{aligned}
&\textit{massimizzare} && 14x_1 + 5x_2 + 4x_3 \\
&\textit{tenuto conto che} && \\
& && 10x_1 + 30x_2 + 20x_3 \leq 5100 \\
& && 10x_1 + 20x_2 + 40x_3 \leq 8000 \\
& && 30x_1 + 10x_2 + 5x_3 \leq 1805 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0 \\
& && x_3 \geq 0
\end{aligned}
$$

Come si vede, abbiamo a che fare con un problema di PL. Per completezza notiamo che ulteriori vincoli che si potrebbero introdurre sono quelli di interezza delle variabili x_1, x_2, x_3: il numero di pacchi realizzati di ciascun tipo deve essere un valore intero. In tal caso ci troveremmo di fronte ad un problema di PLI. Qui però tralasceremo tali vincoli, ammettendo quindi anche la realizzazione di un numero frazionario di pacchi di ciascun tipo.

Concludiamo questa parte osservando che tra gli strumenti a disposizione del ricercatore operativo ci sono anche i *linguaggi di modellizzazione*. Il modello su carta, scritto in linguaggio matematico, può essere ulteriormente tradotto in un linguaggio di modellizzazione e, come vedremo più avanti con l'introduzione di uno di tali linguaggi (si veda l'Appendice B), questo può semplificare notevolmente il lavoro del ricercatore operativo.

Individuazione di un algoritmo di risoluzione

Una volta che abbiamo a disposizione il modello matematico del problema *non* abbiamo ancora risolto il problema stesso, lo abbiamo semplicemente tradotto in un altro linguaggio. Ma perché allora abbiamo fatto questa traduzione? Sostanzialmente perché una volta trasportato il nostro problema reale nel mondo astratto della matematica, possiamo utilizzare tutti gli strumenti che ci fornsice questa disciplina per studiarlo. Questo vuol dire che possiamo studiare la teoria dei modelli matematici e attraverso questa, arivare infine alla definizione di *algoritmi di risoluzione* per essi, ovvero procedure che ricevono in input i modelli e ne restituiscono le soluzioni.
Come si diceva, i modelli di programmazione matematica includono moltissime sottoclassi che possono differire parecchio tra loro per quanto riguarda le proprietà della funzione obiettivo, le proprietà delle funzioni che definiscono i vincoli, il numero e la natura (continue o discrete) delle variabili. Tali differenze implicano a loro volta differenze per quanto riguarda sia la complessità dei problemi che gli algoritmi di risoluzione. Quindi, una volta costruito il modello matematico, occorre riconoscere a quale classe appartiene e di conseguenza, scegliere

un opportuno algoritmo di risoluzione. Durante il corso avremo modo di presentare algoritmi per diversi problemi, con una particolare attenzione per i già citati problemi di PL e PLI.

Validazione del modello

Quando applichiamo l'algoritmo di risoluzione otteniamo *la solzuione del modello matematico*. È bene rimarcare che questa può non coincidere con la soluzione del problema di decisione. Infatti, non dobbiamo dimenticare che un modello è una *rappresentazione* della realtà, non la realtà stessa. Tale rappresentazione potrebbe non essere aderente alla realtà. Per esempio, potremmo aver dimenticato una qualche variabile di decisione e/o un qualche vincolo del problema. Occorre quindi sempre effettuare, anche dopo aver risolto un modello, un'analisi critica del modello stesso e capire se sia o meno necessario correggerlo. Nel caso sia necessario, si dovrà tornare a risolvere il modello aggiornato.

Durante il corso non ci occuperemo oltre di questa fase detta di *validazione del modello*, ma teniamo sempre presente che è un'operazione importante da compiere.

Capitolo 2

Modelli

In questo capitolo ci occuperemo della creazione di modelli, concentrando la nostra attenzione, come già sottolineato in precedenza, su modelli di Programmazione Lineare (PL) e Programmazione Lineare Intera (PLI). Creare un modello matematico vuol dire sostanzialmente fare una traduzione di un problema di decisione, di cui si ha una descrizione a parole, nel linguaggio matematico: le varie componenti del problema di decisione vengono tradotte in oggetti matematici come insiemi, numeri, variabili, equazioni e/o disequazioni, funzioni matematiche. Non esiste una teoria dei modelli matematici e non è possibile pensare di automatizzare l'operazione di creazione di un modello. Il modo migliore per apprendere come creare modelli matematici è attraverso la pratica. Tuttavia, alcune cose ritornano spesso nella creazione di modelli ed è possibile darne una descrizione formale. In questo capitolo daremo prima ampio spazio alla trattazione delle variabili binarie, particolarmente importanti nella creazione di modelli di problemi di decisione. Poi ci occuperemo di non linearità eliminabili, ovvero di funzioni non lineari che possono apparire in certi modelli ma che possono essere sostituite da equivalenti espressioni lineari, in modo da ricondursi a problemi di PL o PLI. Infine introdurremo una serie di problemi di particolare rilevanza nelle applicazioni pratiche. Di ognuno di questi daremo sempre il modello matematico e, in molti casi, anche il modello dello stesso nel linguaggio AMPL (si veda l'Appendice B).

2.1 Variabili binarie

Nel contesto della modellizzazione un ruolo di primo piano è ricoperto dalle variabili binarie. Tali variabili possono assumere due soli valori (convenzionalmente fissati a 0 e 1) e vengono utilizzate nei problemi di decisione quando, come spesso accade, si deve scegliere se effettuare o non effettuare una determinata azione, se un sistema si debba trovare o meno in un determinato stato. Nel seguito si introdurranno diversi casi in cui si fa uso di tali variabili.

2.1.1 Uso di variabili binarie per imporre limitazioni su altre variabili

Supponiamo che nel nostro problema di decisione una certa variabile x abbia una limitazione superiore pari a B *se ci si trova in uno tra due possibili stati*. La scelta tra i due possibili stati viene modellata con una variabile binaria δ e possiamo imporre che lo stato relativo a $\delta = 1$ sia quello per cui x non può superare B. In altre parole, abbiamo la seguente relazione tra δ e x

$$\delta = 1 \quad \Rightarrow \quad x \leq B.$$

Come possiamo modellare tale vincolo logico tramite una disequazione? Una possibilità è la seguente

$$x \leq B\delta + (1 - \delta)M, \tag{2.1}$$

dove M è un limite superiore esplicito o implicito (ovvero derivato da altri vincoli del problema) sui valori che possono essere assunti da x *indipendentemente dallo stato del sistema* (in prima analisi possiamo anche pensare a $M = +\infty$).

Esempio 4 *Un certo impianto di produzione, che ha una capacità produttiva (massimo numero di prodotti realizzabili in una giornata) in condizioni normali pari a Cap_1, può essere fatto funzionare con una capacità ridotta $Cap_2 < Cap_1$. In questo caso i due stati sono il funzionamento normale ($\delta = 0$) o ridotto ($\delta = 1$) dell'impianto e se indichiamo con x il numero di prodotti realizzati in una giornata, possiamo imporre il vincolo (2.1) con $B = Cap_2$ e $M = Cap_1$.*

Esempio 5 *Se non abbiamo limiti dal di sopra espliciti per la variabile x ma sappiamo per esempio che nel problema sono presenti i vincoli*

$$x + y + z \leq 100, \quad x, y, z \geq 0,$$

un limite implicito per x è 100 e possiamo utilizzare tale valore come quantità M.

In modo sostanzialmente analogo si possono utilizzare variabili binarie per modellare le situazioni in cui una certa variabile x abbia una limitazione inferiore pari ad A *se ci si trova in uno tra due possibili stati*. Di nuovo la scelta tra i due possibili stati viene modellata con una variabile binaria δ e possiamo imporre che lo stato relativo a $\delta = 1$ sia quello per cui x non può essere inferiore ad A. In altre parole, abbiamo la seguente relazione tra δ e x

$$\delta = 1 \quad \Rightarrow \quad x \geq A. \tag{2.2}$$

Il vincolo che lega δ e x è il seguente

$$x \geq A\delta - (1 - \delta)M,$$

dove $-M$ è un limite inferiore esplicito o implicito sui valori che possono essere assunti da x *indipendentemente dallo stato del sistema*. In particolare, se abbiamo un vincolo di non negatività per x possiamo imporre $M = 0$. Si noti che un modo per imporre la relazione (2.2) è anche

$$\delta x \geq \delta A.$$

Tuttavia, l'inconveniente di questa disequazione è la perdita di linearità: δx non è un'espressione lineare.

2.1.2 Uso di variabili binarie per imporre vincoli

In alcuni problemi può accadere che un certo vincolo $\sum_{j=1}^{n} a_j x_j \leq b$ sia presente solo se un sistema si trova in uno tra due possibili stati, identificato, ad esempio, dal valore 1 di una variabile binaria δ. In altre parole si ha

$$\delta = 1 \quad \Rightarrow \quad \sum_{j=1}^{n} a_j x_j \leq b.$$

Possiamo modellare questa implicazione con la seguente disequazione

$$\sum_{j=1}^{n} a_j x_j \leq b\delta + M(1-\delta),$$

dove M è un numero sufficentemente elevato, tale da rendere la disequazione $\sum_{j=1}^{n} a_j x_j \leq M$ (a cui ci si riduce nel caso $\delta = 0$) ridondante rispetto agli altri vincoli del problema. Per esempio, se sono note delle limitazioni inferiori l_j e superiori u_j per tutte le variabili x_j, una possibile scelta per M è la seguente

$$M = \sum_{j=1}^{n} \max\{a_j l_j, a_j u_j\}.$$

Esempio 6 *Supponiamo che i e j siano due attività di durata rispettivamente pari a d_i e d_j che non possano essere eseguite contemporaneamente. Associamo alle due attività due variabili t_i e t_j che indicano il loro istante di inizio. Ipotiizziamo anche che le attività debbano essere iniziate in un determinato intervallo, ovvero che esistano istanti $T_{\min}$ e $T_{\max}$ tali che*

$$T_{\min} \leq t_i, t_j \leq T_{\max}.$$

Se le dua attività non possono essere eseguite contemporaneamente possiamo introdurre una variabile binaria

$$\delta_{ij} = \begin{cases} 0 & \textit{se } i \textit{ precede } j \\ 1 & \textit{se } j \textit{ precede } i \end{cases}$$

In tal caso avremo

$$\delta_{ij} = 0 \quad \Rightarrow \quad t_j \geq t_i + d_i \quad (j \textit{ può iniziare solo quando finisce } i)$$

$$\delta_{ij} = 1 \quad \Rightarrow \quad t_i \geq t_j + d_j \quad (i \textit{ può iniziare solo quando finisce } j)$$

In base a quanto visto le due implicazioni possono essere tradotte nei seguenti vincoli

$$t_j \geq t_i + d_i(1-\delta_{ij}) - M\delta_{ij}$$
$$t_i \geq t_j + d_j\delta_{ij} - M(1-\delta_{ij}),$$

dove possiamo scegliere $M = T_{\max} - T_{\min}$.

2.1.3 Costi fissi

Le variabili binarie vengono frequentemente usate per modellare problemi in cui sono presenti costi fissi. Pensiamo al caso di una variabile x che rappresenta la quantità realizzata di un certo prodotto. Se $x = 0$ avremo ovviamente un costo di produzione associato al prodotto pari a 0. Ma se $x > 0$ allora avremo un costo pari a $f + cx$ dove c è il costo di produzione per unità di prodotto e f è un costo fisso (legato, ad esempio, al fatto che la produzione richiede l'acquisto di un certo macchinario il cui costo è, appunto, fisso e non dipende dalla quantità prodotta). In tal caso si introduce una variabile binaria

$$\delta = \begin{cases} 0 & \text{se } x = 0 \\ 1 & \text{se } x > 0 \end{cases} \tag{2.3}$$

grazie alla quale possiamo scrivere il costo come

$$cx + f\delta \tag{2.4}$$

Abbiamo bisogno di modellare l'implicazione

$$\delta = 0 \quad \Rightarrow \quad x = 0. \tag{2.5}$$

Se M è un limite noto (implicito o esplicito) per i valori che possono essere assunti da x (si veda anche la discussione nella Sezione 2.1.1) possiamo imporre

$$x \leq M\delta,$$

che combinata con il vincolo di non negatività $x \geq 0$ (la produzione non può ovviamente essere negativa), garantisce che (2.5) sia soddisfatta. In realtà per soddisfare (2.3) dovremmo anche imporre

$$\delta = 1 \quad \Rightarrow \quad x > 0. \tag{2.6}$$

Questo non viene imposto ma è in realtà una condizione sempre soddisfatta dalle *soluzioni ottime* del problema. Infatti, il costo (2.4) comparirà in un obiettivo da minimizzare

$$\begin{aligned} \min \quad & \cdots + (f\delta + cx) + \cdots \\ & \vdots \\ & x \leq M\delta \\ & \vdots \\ & x \geq 0 \end{aligned}$$

La combinazione $\delta = 1, x = 0$, che viola (2.6) può comparire in una soluzione ammissibile del problema, ma certamente tale soluzione non sarà ottima, in quanto basta portare il valore di δ a 0 per ridurre di f il valore dell'obiettivo.

2.1.4 Vincoli logici

Spesso accade che esistano dei vincoli logici che legano i valori di diverse variabili binarie. Ad esempio, ipotizziamo di avere quattro attività A, B, C, D che possiamo decidere se svolgere o non svolgere e che valga il seguente vincolo:

se si esegue A o B, allora si esegue C o non si esegue D

(gli *o* vanno intesi come non esclusivi). Indichiamo con V_i, $i = A, B, C, D$, l'evento *si esegue l'attività i*. Utilizzando gli operatori logici $\cup$ (OR), $\cap$ (AND), $\neg$ (NOT), $\Rightarrow$ (implicazione), possiamo scrivere il vincolo come

$$V_A \cup V_B \quad \Rightarrow \quad V_C \cup \neg V_D.$$

Ricordando alcuni risultati sulle operazioni logiche come

$$\begin{array}{lcl} S_1 \;\Rightarrow\; S_2 & \equiv & \neg S_1 \cup S_2 \\ \neg(S_1 \cup S_2) & \equiv & \neg S_1 \cap \neg S_2 \\ \neg(S_1 \cap S_2) & \equiv & \neg S_1 \cup \neg S_2 \\ S_1 \cup (S_2 \cap S_3) & \equiv & (S_1 \cup S_2) \cap (S_1 \cup S_3) \\ S_1 \cap (S_2 \cup S_3) & \equiv & (S_1 \cap S_2) \cup (S_1 \cap S_3) \end{array}$$

è possibile mostrare come ogni espressione logica che coinvolge gli operatori $\cup$, $\cap$, $\neg$, $\Rightarrow$ può essere riscritta in *forma normale disgiuntiva*

$$\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_k \cup \neg\mathcal{E}_{k+1} \cup \cdots \cup \neg\mathcal{E}_{k+h},$$

dove ogni $\mathcal{E}_i$, $i = 1, \ldots, k+h$ è una espressione data dall'intersezione di un numero finito di eventi (eventualmente negati), oppure in *forma normale congiuntiva*

$$\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_k \cap \neg\mathcal{E}_{k+1} \cap \cdots \cap \neg\mathcal{E}_{k+h},$$

dove ogni $\mathcal{E}_i$, $i = 1, \ldots, k+h$ è una espressione data dall'unione di un numero finito di eventi (eventualmente negati). Nel nostro esempio abbiamo la forma normale disgiuntiva

$$\underbrace{(\neg V_A \cap \neg V_B)}_{\mathcal{E}_1} \cup \underbrace{V_C}_{\mathcal{E}_2} \cup \underbrace{\neg V_D}_{\mathcal{E}_3}, \tag{2.7}$$

oppure la forma normale congiuntiva

$$\underbrace{(\neg V_A \cup V_C \cup \neg V_D)}_{\mathcal{E}_4} \cap \underbrace{(\neg V_B \cup V_C \cup \neg V_D)}_{\mathcal{E}_5}. \tag{2.8}$$

Se ora introduciamo le variabili binarie

$$\delta_i = \begin{cases} 0 & \text{se si decide di non eseguire l'attività } i \\ 1 & \text{altrimenti} \end{cases}$$

$i = A, B, C, D$, vediamo come una forma normale congiuntiva e disgiuntiva può essere tradotta in un sistema di disequazioni lineari che coinvolge queste

variabili binarie (più altre eventualmente da aggiungere). Ogni OR di eventi (eventualmente negati)

$$V_1 \cup \cdots \cup V_k \cup \neg V_{k+1} \cup \cdots \cup \neg V_{k+h},$$

a cui si associano le variabili binarie δ_i, $i =, \ldots, k+h$, viene tradotto nella seguente disequazione

$$\sum_{i=1}^{k} \delta_i + \sum_{i=k+1}^{k+h} (1 - \delta_i) \geq 1$$

(almeno una della variabili δ_i, $i = 1, \ldots, k$, deve essere pari a 1 oppure almeno una delle variabili δ_i, $i = k+1, \ldots, k+h$, deve essere pari a 0). Se si ha invece un AND di eventi (eventualmente negati)

$$V_1 \cap \cdots \cap V_k \cap \neg V_{k+1} \cap \cdots \cap \neg V_{k+h},$$

allora possiamo introdurre, oltre alle variabili binarie δ_i, $i =, \ldots, k+h$, un'ulteriore variabile binaria δ il cui valore pari a 1 implica che l'espressione è soddisfatta, ovvero

$$\delta = 1 \quad \Rightarrow \quad \delta_i = 1 \;\; i = 1, \ldots, k, \quad \delta_i = 0 \;\; i = k+1, \ldots, k+h,$$

o, equivalentemente

$$\begin{array}{ll} \delta_i \geq \delta & i = 1, \ldots, k \\ \delta_i \leq 1 - \delta & i = k+1, \ldots, k+h. \end{array}$$

A questo punto vediamo come tradurre (2.7) e (2.8). In (2.7) cominciamo con l'introdurre la variabile δ in rappresentanza del'espressione $\mathcal{E}_1$ imponendo

$$\delta_A \leq 1 - \delta, \quad \delta_B \leq 1 - \delta,$$

dopodiché possiamo scrivere l'OR come

$$\delta + \delta_C + (1 - \delta_D) \geq 1.$$

Quindi, complessivamente (2.7) equivale al sistema di vincoli lineari

$$\left\{ \begin{array}{l} \delta_A \leq 1 - \delta \\ \delta_B \leq 1 - \delta \\ \delta + \delta_C - \delta_D \geq 0. \end{array} \right.$$

Per quanto riguarda (2.8) abbiamo invece che $\mathcal{E}_4$ equivale a

$$\delta_C + (1 - \delta_A) + (1 - \delta_D) \geq 1,$$

mentre $\mathcal{E}_5$ equivale a

$$(1 - \delta_B) + \delta_C + (1 - \delta_D) \geq 1,$$

da cui (2.8) equivale al sistema di vincoli lineari

$$\begin{cases} \delta_A + \delta_D - \delta_C \leq 1 \\ \delta_B + \delta_D - \delta_C \leq 1. \end{cases}$$

Nel caso specifico le due formulazioni ottenute tramite la forma congiuntiva e quella disgiuntiva sono tra loro equivalenti, ma da un punto di vista algoritmico si osserva che la forma disgiuntiva è spesso migliore rispetto a quella congiuntiva.

NOTA BENE L'AND di due eventi S_1 e S_2 con associate le variabili binarie δ_1 e δ_2, oltre a poter essere modellato, con l'introduzione della variabile binaria aggiuntiva δ, come

$$\delta_1 \geq \delta, \quad \delta_2 \geq \delta$$

può essere modellato anche con un vincolo come

$$\delta_1 \delta_2 \geq \delta.$$

Tuttavia, un vincolo di questo tipo è non lineare ed è opportuno quindi evitarne l'introduzione.

2.2 Non linearità eliminabili

Per quanto la presenza di espressioni lineari in un modello sia sempre auspicabile per la maggiore facilità di risoluzione dei problemi lineari, non sempre è possibile evitare l'introduzione di espressioni non lineari. Per esempio, prendiamo la semplicissima formula della velocità in un moto rettilineo uniforme

$$v = \frac{s}{t}$$

dove v indica la velocità, s lo spazio percorso e t il tempo. Se supponiamo che queste siano tre variabili di un problema di decisione, è chiaro che il vincolo dato dalla formula che lega le tre grandezze è non lineare e non possiamo rimuovere tale non linearità.

Esistono però anche casi in cui la non linearità può essere eliminata con l'introduzione di opportune espressioni lineari. Vediamo di seguito un paio di esempi.

2.2.1 Problemi maximin e minimax

Si consideri un problema di questo tipo (problema minimax in cui si minimizza il massimo di un numero finito di funzioni lineari)

$$\begin{array}{lll} \min & \max_{r=1,\ldots,k}\{\sum_{j=1}^{n} c_{rj}x_j + c_{0r}\} & \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1,\ldots,m \\ & x_j \geq 0 & j = 1,\ldots,n \end{array} \tag{2.9}$$

La funzione obiettivo

$$f(x) = \max_{r=1,\dots,k} \{\sum_{j=1}^{n} c_{rj}x_j + c_{0r}\}$$

è non lineare. Tuttavia, si può vedere che (2.9) è equivalente al seguente problema di programmazione lineare

$$\begin{array}{lll} \min & y & \\ & y \geq \sum_{j=1}^{n} c_{rj}x_j + c_{0r} & r = 1,\dots,k \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1,\dots,m \\ & x_j \geq 0 & j = 1,\dots,n \end{array}$$

Un discorso analogo vale per la massimizzazione del minimo di un numero finito di funzioni lineari (problema maximin), mentre si può verificare che non è eliminabile la non linearità nei problemi di massimizzazione del massimo di un numero finito di funzioni lineari (maximax) e minimizzazione del minimo di un numero finito di funzioni lineari (minimin). Teniamo comunque presente come problemi maximax e problemi minimin siano risolvibili risolvendo più problemi di PL. Ad esempio, se abbiamo il problema minimin

$$\begin{array}{lll} \min & \min_{r=1,\dots,k}\{\sum_{j=1}^{n} c_{rj}x_j + c_{0r}\} & \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1,\dots,m \\ & x_j \geq 0 & j = 1,\dots,n \end{array}$$

lo possiamo risolvere risolvendo i k problemi di PL per $r = 1,\dots,k$:

$$\begin{array}{lll} \min & \sum_{j=1}^{n} c_{rj}x_j + c_{0r} & \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1,\dots,m \\ & x_j \geq 0 & j = 1,\dots,n \end{array}$$

2.2.2 Minimizzazione della somma di valori assoluti di funzioni lineari

Si consideri il problema

$$\begin{array}{lll} \min & \sum_{r=1}^{k} \mid \sum_{j=1}^{n} c_{rj}x_j + c_{0r} \mid & \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1,\dots,m \\ & x_j \geq 0 & j = 1,\dots,n \end{array} \tag{2.10}$$

La funzione obiettivo

$$f(x) = \sum_{r=1}^{k} \mid \sum_{j=1}^{n} c_{rj}x_j + c_{0r} \mid$$

è non lineare. In realtà questo caso è riconducibile al precedente osservando che

$$\mid \sum_{j=1}^{n} c_{rj}x_j + c_{0r} \mid = \max\{\sum_{j=1}^{n} c_{rj}x_j + c_{0r}, -\sum_{j=1}^{n} c_{rj}x_j - c_{0r}\}.$$

Abbiamo quindi che (2.10) è equivalente a

$$\begin{array}{lll} \min & \sum_{r=1}^{k} \max\{\sum_{j=1}^{n} c_{rj}x_j + c_{0r}, -\sum_{j=1}^{n} c_{rj}x_j - c_{0r}\} & \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n \end{array}$$

che a sua volta è equivalente al problema lineare

$$\begin{array}{lll} \min & \sum_{j=1}^{k} y_j & \\ & y_j \geq \sum_{j=1}^{n} c_{rj}x_j + c_{0r} & r = 1, \ldots, k \\ & y_j \geq -\sum_{j=1}^{n} c_{rj}x_j - c_{0r} & r = 1, \ldots, k \\ & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n \end{array}$$

2.3 Problemi vari e relativi modelli

In questa sezione introdurremo diversi problemi di decisione di cui ricaveremo il modello matematico e, in molti casi, anche quello in linguaggio AMPL (si veda l'Appendice B).

2.3.1 Problemi di flusso a costo minimo

Si consideri un'azienda che realizza un certo prodotto ed è formata da diversi centri che si distinguono in

- centri che fungono da magazzini, in cui il prodotto si limita a transitare;
- centri di produzione, in cui, oltre a transitare, il prodotto viene anche realizzato in quantità prefissate;
- centri di distribuzione, in cui, oltre a transitare, il prodotto viene anche rivenduto in quantità prefissate.

A ogni centro i si associa un valore intero b_i che sarà:

- $= 0$ per i magazzini;
- > 0 per i centri di produzione (b_i rappresenta la quantità di prodotto realizzata in quel centro);
- < 0 per i centri di distribuzione ($-b_i$ rappresenta la quantità di prodotto rivenduta in quel centro).

Si suppone che $\sum b_i = 0$, ovvero che la quantità di prodotto totale realizzata nei vari centri di produzione sia esattamente pari a quella complessivamente rivenduta in tutti i centri di distribuzione. Alcuni centri dell'azienda comunicano tra loro tramite linee di collegamento alle quali sono associati dei costi di trasporto:

se abbiamo una linea di collegamento tra il centro i e il centro j, indicheremo con c_{ij} il costo di trasporto di una singola unità di prodotto lungo tale linea. In alcuni casi alla linea è anche associato un altro valore, la sua capacità d_{ij}, che indica la massima quantità di prodotto trasportabile lungo essa. Il problema del *flusso a costo minimo* consiste nel determinare come instradare il prodotto all'interno della rete di collegamento dell'azienda in modo che il prodotto realizzato nei centri di produzione giunga ai centri di distribuzione a un costo totale di trasporto complessivo minimo.

Vale la pena osservare che, pur avendo qui noi discusso il caso di una rete di collegamento tra centri di un'azienda, un problema di questo tipo può sorgere in altri tipi di rete, come una rete di comunicazione (dove il prodotto trasportato è informazione, i centri sono computer e le linee di collegamento connessioni tra computer), oppure come una rete idraulica (dove il prodotto trasportato è acqua, dove abbiamo centri di immissione dell'acqua nella rete, equivalenti ai centri di produzione, centri di smistamento dell'acqua nella rete, equivalenti ai magazzini e centri di distribuzione dell'acqua, dove, infine, le linee di collegamento sono tubi). Per slegarci dai dettagli non significativi delle applicazioni da cui sorgono questi problemi, possiamo utilizzare i grafi (si veda l'Appendice A). La rete (sia essa quella dell'azienda, quella di comunicazione, quella idraulica o quella proveniennte da una qualche altra applicazione) viene rappresentata con un grafo orientato $G = (V, A)$ dove i nodi corrispondono ai vari centri e gli archi alle linee di collegamento tra essi. Ai nodi in V vengono associati i valori b_i relativi ai centri che rappresentano. Sulla base di tali valori i nodi $i \in V$ vengono suddivisi in tre categorie:

- nodi *sorgente* (quelli con valore $b_i > 0$);
- nodi *destinazione* ($b_i < 0$);
- nodi *transito* ($b_i = 0$).

Agli archi $(i, j) \in A$ sono associati i costi unitari c_{ij} ed eventualmente anche i valori di capacità d_{ij} relativi alle linee di collegamento che rappresentano.

Modello matematico del problema di flusso a costo minimo

Di seguito ricaveremo il modello matematico del problema.

Variabili Associamo una variabile x_{ij} ad ogni arco (i, j) della rete:

$$(i, j) \in A \quad \rightarrow \quad x_{ij} = \text{quantità prodotto inviata lungo l'arco } (i, j)$$

Per tali variabili sarà richiesto che:

- siano ≥ 0 (non si possono inviare quantità negative di prodotto lungo gli archi);
- siano $\leq d_{ij}$ (limite di capacità massima degli archi, se presente);

- siano intere (se il prodotto non è frazionabile).

Vincoli In ogni nodo $i \in V$ si deve avere:

$$\text{(Flusso uscente da } i\text{) - (Flusso entrante in } i\text{)} = b_i$$

Infatti: se il nodo è di transito, questo vincolo dice che la quantità di prodotto uscente da i è esattamente pari a quella entrante; se il nodo è sorgente, questo vincolo dice che la quantità di prodotto uscente da i è pari a quella entrante in i sommata alla quantità di prodotto b_i realizzata in i; se il nodo è destinazione, questo vincolo dice che la quantità di prodotto entrante in i è pari a quella uscente da i sommata alla quantità di prodotto $-b_i$ che viene rivenduta in i. Vediamo ora di ricavare le espressioni matematiche per flusso uscente e flusso entrante in i. Flusso uscente da i:

$$\sum_{j:(i,j)\in A} x_{ij}$$

Flusso entrante in i:

$$\sum_{j:(j,i)\in A} x_{ji}$$

Quindi, i vincoli corrisponderanno alle seguenti equazioni:

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall\, i \in V$$

Obiettivo Se il costo di trasporto di un'unità di prodotto lungo (i, j) è c_{ij}, il costo di trasporto di x_{ij} unità di prodotto è $c_{ij}x_{ij}$. Se sommiamo su tutti gli archi otteniamo il costo totale di trasporto:

$$\sum_{(i,j)\in A} c_{ij}x_{ij}$$

che vorremo minimizzare.

Il modello matematico completo è quindi il seguente:

$$\begin{array}{ll} \min \sum_{(i,j)\in A} c_{ij}x_{ij} & \\ \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i & \forall\, i \in V \\ 0 \le x_{ij} \le d_{ij} \text{ interi} & \forall\, (i,j) \in A \end{array}$$

Chiamiamo matrice dei vincoli di uguaglianza di tale problema la matrice avente tante righe quanti sono i vincoli (e quindi tante quante sono i nodi della rete) e tante colonne quante sono le variabili del problema (e quindi tanti quanti sono gli archi della rete). Vale la seguente osservazione:

Osservazione 1 *La matrice dei vincoli di uguaglianza per i problemi di flusso a costo minimo coincide con la matrice di incidenza nodo-arco della rete.*

Dimostrazione La colonna della matrice relativa alla variabile x_{ij} corrisponde a un arco $(i,j) \in A$. Abbiamo che x_{ij} ha coefficiente +1 nel vincolo relativo al nodo i (essendo l'arco (i,j) uscente da i, x_{ij} appare in $\sum_{j:(i,j)\in A} x_{ij}$), coefficiente -1 nell'equazione relativa al nodo j (essendo l'arco (i,j) entrante in j, x_{ij} appare in $-\sum_{i:(i,j)\in A} x_{ij}$), mentre ha coefficiente 0 in corrispondenza delle equazioni relative a tutti gli altri nodi, esattamente come la matrice di incidenza nodo-arco della rete nella colonna relativa proprio all'arco (i,j).

Esempio 7 *Sia data la rete in Figura 2.1. I valori b_i sono riportati di fianco*

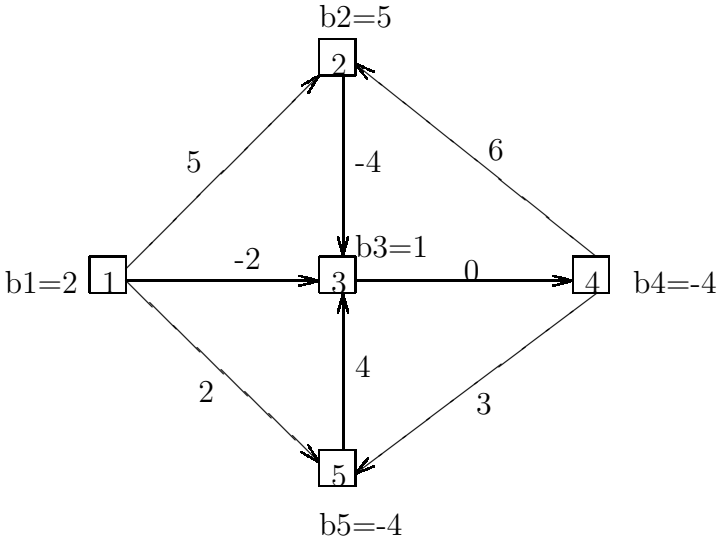


Figura 2.1: Una rete con i relativi valori b_i associati ai nodi ed i costi unitari di trasporto lungo gli archi.

ai nodi mentre lungo gli archi sono riportati i valori c_{ij} (non sono imposti in questo caso limiti di capacità d_{ij} sugli archi) . I nodi 1,2 e 3 sono nodi sorgente mentre i nodi 4 e 5 sono nodi destinazione (non vi sono nodi transito). Il problema corrispondente è il seguente

$$\min\ 5x_{12} - 4x_{23} + 6x_{42} - 2x_{13} + 0x_{34} + 2x_{15} + 4x_{53} + 3x_{45}$$
$$x_{12} + x_{13} + x_{15} = 2$$
$$x_{23} - x_{12} - x_{42} = 5$$
$$x_{34} - x_{13} - x_{23} - x_{53} = 1$$
$$x_{42} + x_{45} - x_{34} = -4$$
$$x_{53} - x_{15} - x_{45} = -4$$
$$x_{12}, x_{23}, x_{42}, x_{13}, x_{34}, x_{15}, x_{53}, x_{45} \geq 0 \text{ interi}$$

La matrice dei vincoli di uguaglianza per questo esempio è la seguente:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

e si vede che coincide con la matrice di incidenza nodo-arco della rete.

Modello AMPL

Vediamo ora il modello AMPL per il problema di flusso a costo minimo.

MIN-FLOW.MOD

INSIEMI

set $NODI$ **;**
set $ARCHI$ **within** $NODI$ **cross** $NODI$ **;**

PARAMETRI

param $b\{NODI\}$**;**
param $c\{ARCHI\}$ **;**
param $d\{ARCHI\} >= 0$, **default** Infinity **;**

VARIABILI

var $x\{(i,j)$ **in** $ARCHI\} >= 0$, <= d[i,j], **integer ;**

VINCOLI

subject to bilancio$\{i$ **in** $NODI\}$ **:** sum$\{j$ **in** $NODI : (i,j)$ **in** $ARCHI\}$ x[i,j]
- sum$\{j$ **in** $NODI : (j,i)$ **in** $ARCHI\}$ x[j,i] = b[i] **;**

OBIETTIVO

minimize costo_totale **:** sum$\{(i,j)$ **in** $ARCHI\}$ c[i,j]*x[i,j] **;**

Vediamo di commentare alcune parti di questo modello. Nella dichiarazione dell'insieme $ARCHI$ abbiamo la parola chiave **within**. Questa sta a indicare che l'insieme dichiarato è sottinisieme dell'insieme che segue. Quindi, in generale se dichiariamo

set A **within** B **;**

questo ci dice che l'insieme A è un sottinsieme di B (se al momento della definizione di A e B questa condizione non fosse soddisfatta, verrebbe segnalato un errore). Sempre nella dichiarazione di $ARCHI$ abbiamo la parola chiave **cross**. Questa definisce il prodotto cartesiano tra insiemi. Quindi, in generale avremo che A **cross** B è il prodotto cartesiano degli insiemi A e B, ovvero l'insieme di tutte le coppie con il primo elemento in A e il secondo in B. Dalla dichiarazione di $ARCHI$ ricaviamo dunque che $ARCHI$ è un sottinsieme del prodotto cartesiano dell'insieme $NODI$ con se stesso. L'operazione di prodotto cartesiano è tra quelle che possono essere eseguite tra insiemi. Tra le altre, citiamo anche l'operazione di differenza tra insiemi A **diff** B (differenza tra l'insieme A e l'insieme B, ovvero l'insieme che contiene tutti gli elementi che sono in A ma non in B).

Nella dichiarazione dela parametro d abbiamo la parola chiave **default**. Questa viene utilizzata per indicare i valori di default di un parametro: per tutti i casi in cui non si specifica il valore del paraemetro questo viene fissato al valore di default. Nel nostro caso il valore del parametro d verrà fissato a ∞ ogni volta che questo non sarà definito esplicitamente nel file di dati.

Nella dichiarazione dei vincoli abbiamo, ad esempio

sum$\{j$ **in** $NODI : (i, j)$ **in** $ARCHI\}$ x[i,j]

In pratica, invece di eseguire la somma su tutti gli elemnti dell'insieme $NODI$, la eseguiamo solo su quelli che soddisfano una determinata condizione introdotta da un **:**. È possibile che le condizioni da soddisfare siano anche più di una. In tal caso le si elenca sempre dopo i **:** separate tra loro dalla parola chiave **and**. Infine, osserviamo che la frequente occorrenza di problemi di flusso nella pratica ha spinto a inserire in AMPL una sintassi speciale per tali problemi. Non ci addentreremo in questa ma si rimandano gli interessati alla lettura del manuale.

A questo punto i dati relativi all'esempio li possiamo inserire nel file MINFLOW.DAT (si noti che, non essendo specificate le capacità sugli archi, queste

saranno tutte pari a ∞).

MIN-FLOW.DAT

INSIEMI

set $NODI$ **:=** n1 n2 n3 n4 n5 **;**
set $ARCHI$ **:=** (n1,n2) (n1,n3) (n1,n5) (n2,n3) (n3,n4) (n4,n2) (n4,n5) (n5,n3)
;

PARAMETRI

param b **:=**
n1 2
n2 5
n3 1
n4 -4
n5 -4
;

param c **:=**
n1 n2 5
n1 n3 -2
n1 n5 2
n2 n3 -4
n3 n4 0
n4 n2 6
n4 n5 3
n5 n3 4 **;**

2.3.2 Problemi di flusso massimo

Analizziamo ora un problema simile al precedente ma con le seguenti differenze:

- l'azienda ha un solo centro di produzione e uno solo di distribuzione, tutti gli altri centri sono magazzini;
- non abbiamo costi di trasporto unitari lungo le linee di collegamento ma abbiamo sempre limiti di capacità su di esse;

- la quantità di prodotto realizzata nell'unico centro di produzione (equivalente a quella ricevuta dall'unico centro di distribuzione) non è fissata a priori ma è variabile.

Il problema di flusso massimo consiste nello stabilire qual è la quantità massima di prodotto che può essere realizzata dal centro di produzione, instradata attraverso la rete di collegamento e fatta giungere al centro di distribuzione, tenendo conto dei limiti di capacità sulle linee di collegamento. Naturalmente anche qui si può pensare ad altri contesti applicativi per tale problema, quali reti di computer e reti idrauliche come nel problema di flusso a costo minimo. Inoltre, anche qui possiamo svincolarci dai dettagli delle applicazioni rappresentando il problema tramite un grafo, costruito in modo identico a quello del problema di flusso a costo minimo. In tale grafo il singolo nodo sorgente verrà convenzionalmente indicato con S e il singolo nodo destinazione con D. Teniamo presente che la restrizione a un singolo nodo sorgente e un singolo nodo destinazione può non essere vera in certe applicazioni (nulla impedisce che l'azienda abbia più centri di produzione e/o più centri di distribuzione). In realtà con un piccolo artificio possiamo sempre ricondurci al caso di un solo nodo sorgente e un solo nodo destinazione. Se sono presenti più sorgenti e/o destinazioni, introduciamo una sorgente fittizia collegata tramite archi fittizi a capacità infinita a ciascuna sorgente reale e, analogamente, una destinazione fittizia alla quale si giunge tramite archi fittizi a capacità infinita a partire da ciascuna destinazione reale (vedi Figura 2.2). Le sorgenti e destinazioni reali diventano a questo punto nodi di transito e rimangono solo la singola sorgente fittizia e la singola destinazione fittizia.

Modello matematico del problema

Vediamo ora di introdurre un modello matematico che rappresenti il nostro problema, peraltro molto simile a quello del problema di flusso a costo minimo.

Variabili Associamo ad ogni arco della rete $(i, j) \in A$ una variabile:

$$x_{ij} = \text{flusso inviato lungo l'arco } (i, j)$$

Tali variabili saranno vincolate ad essere non negative (non ha senso parlare di un flusso negativo). Se indichiamo con d_{ij} la capacità dell'arco (i, j) si dovrà anche avere

$$x_{ij} \leq d_{ij} \quad \forall\ (i, j) \in A,$$

cioè il flusso lungo ogni arco non ne può superare la capacità. Per prodotti non frazionabili le variabili possono assumere solo valori interi.

Vincoli I vincoli sono quelli di equilibrio nei nodi intermedi, che sono tutti nodi di transito. Questi possono essere tradotti nelle seguenti equazioni:

$$\underbrace{\sum_{j:\ (k,j)\in A} x_{kj}}_{\text{flusso uscente da } k} = \underbrace{\sum_{j:\ (j,k)\in A} x_{jk}}_{\text{flusso entrante in } k} \qquad \forall\ k \in V \setminus \{S, D\}.$$

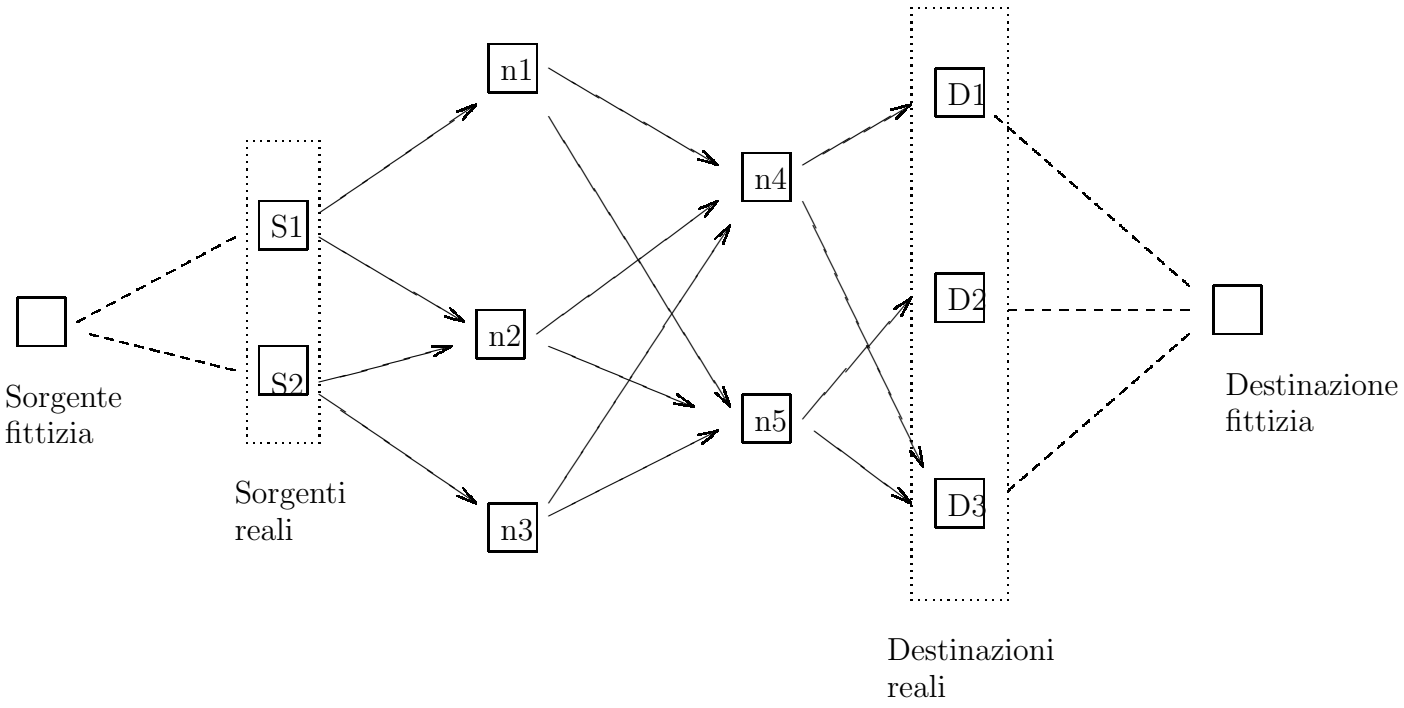


Figura 2.2: La riduzione al caso di una sola sorgente e una sola destinazione.

Obiettivo Il nostro obiettivo è quello di massimizzare la quantità di flusso inviato dal nodo sorgente S e quindi il flusso uscente dal nodo sorgente S:

$$\sum_{j:\ (S,j)\in A} x_{Sj}$$

o, equivalentemente, quella entrante nel nodo destinazione D:

$$\sum_{j:\ (j,D)\in A} x_{jD}$$

(le due quantità sono uguali).

Riassumendo, il modello matematico del problema di massimo flusso è il seguente:

$$\begin{array}{lll} \max & \sum_{j:\ (S,j)\in A} x_{Sj} & \\ & \sum_{j:\ (k,j)\in A} x_{kj} = \sum_{j:\ (j,k)\in A} x_{jk} & \forall\ k \in V \setminus \{S, D\} \\ & 0 \le x_{ij} \le d_{ij} & \forall\ (i,j) \in A \end{array} \qquad (2.11)$$

Se andiamo a considerare la matrice dei vincoli di uguaglianza di questo problema, possiamo, in modo del tutto analogo alla dimostrazione dell'Osservazione 1, dimostrare il seguente risultato.

Osservazione 2 *La matrice dei vincoli di uguaglianza del problema di flusso massimo coincide con la matrice di incidenza nodo-arco della rete senza le righe relative ai nodi S e D.*

Esempio 8 *Sia data la rete in Figura 2.3. I numeri sugli archi ne indicano le*

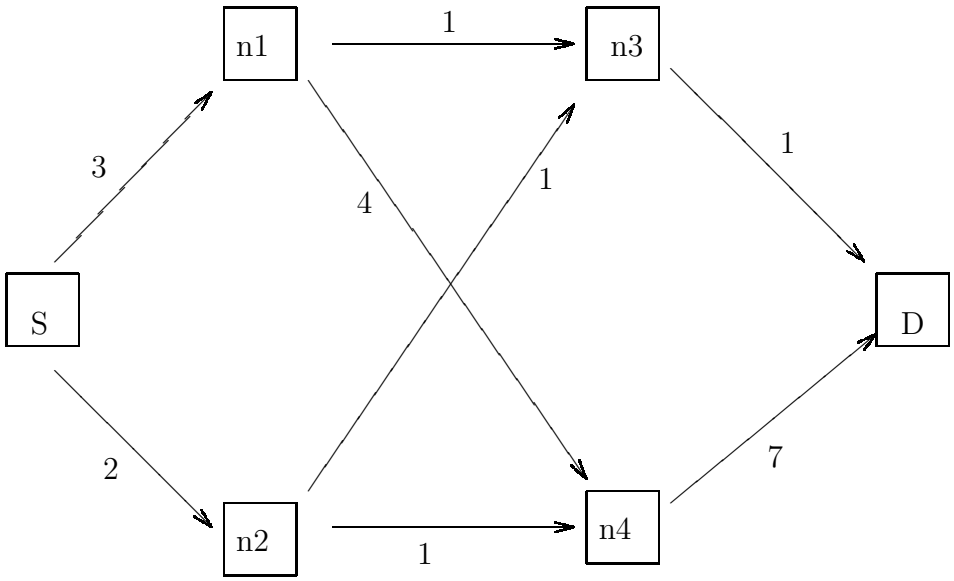


Figura 2.3: Una rete con le capacità degli archi indicati al loro fianco.

capacità. Il modello matematico del problema è il seguente problema di PLI:

$$
\begin{aligned}
\max \quad & x_{S1} + x_{S2} \\
& x_{13} + x_{14} = x_{S1} \\
& x_{23} + x_{24} = x_{S2} \\
& x_{3D} = x_{13} + x_{23} \\
& x_{4D} = x_{14} + x_{24} \\
0 \le x_{S1} \le 3 \quad & \textit{intera} \\
0 \le x_{S2} \le 2 \quad & \textit{intera} \\
0 \le x_{13} \le 1 \quad & \textit{intera} \\
0 \le x_{14} \le 4 \quad & \textit{intera} \\
0 \le x_{23} \le 1 \quad & \textit{intera} \\
0 \le x_{24} \le 1 \quad & \textit{intera} \\
0 \le x_{3D} \le 1 \quad & \textit{intera} \\
0 \le x_{4D} \le 7 \quad & \textit{intera}
\end{aligned}
$$

La matrice dei vincoli di uguaglianza per questo esempio è la seguente:

	$(S,1)$	$(S,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,D)$	$(4,D)$
1	-1	0	1	1	0	0	0	0
2	0	-1	0	0	1	1	0	0
3	0	0	-1	0	-1	0	1	0
4	0	0	0	-1	0	-1	0	1

Modello AMPL

Vediamo ora il modello AMPL per il problema di flusso massimo.

MAX-FLOW.MOD

INSIEMI

set $NODI$ **;**
set $ARCHI$ **within** $NODI$ **cross** $NODI$ **;**

PARAMETRI

param Sorgente **symbolic in** $\{NODI\}$**;**
param Destinazione **symbolic in** $\{NODI\}$, **!=** Sorgente **;**
param $d\{ARCHI\} >= 0$, **default** Infinity **;**

VARIABILI

var $x\{(i,j)$ **in** $ARCHI\} >= 0$, <= d[i,j], **integer ;**

VINCOLI

subject to equilibrio$\{i$ **in** $NODI$ **diff** {Sorgente, Destinazione}$\}$ **:** sum$\{j$ **in** $NODI : (i,j)$ **in** $ARCHI\}$ x[i,j] - sum$\{j$ **in** $NODI : (j,i)$ **in** $ARCHI\}$ x[j,i] = 0 **;**

OBIETTIVO

maximize flusso_uscente **:** sum$\{j$ **in** $NODI :$ (Sorgente,$j)$ **in** $ARCHI\}$ x[Sorgente,j]

;

Il modello è molto simile a quello di flusso a costo minimo ma in esso si deve fronteggiare un problema. Per definire i vincoli e l'obiettivo del problema abbiamo bisogno di identificare un nodo sorgente e uno destinazione. Ma al momento della stesura del modello non sappiamo quale nodo della rete sarà quello sorgente e quale quello destinazione (non solo, tali nodi saranno ovviamente diversi in varie istanze del problema). Più precisamente, al momento della stesura del modello non conosciamo proprio quali siano i nodi delle rete che verranno specificati solo nel file di dati. Per aggirare questa difficoltà si usa la sintassi

param Sorgente **symbolic in** $\{NODI\}$**;**
param Destinazione **symbolic in** $\{NODI\}$, **!=** Sorgente **;**

Con questa dichiarazione diciamo che *esiste* un elemento dell'insieme $NODI$ (**in** $NODI$) che identifichiamo con il simbolo (**symbolic**) Sorgente e un elemento dell'insieme $NODI$ che identifichiamo con il simbolo Destinazione diverso (**!=**) da Sorgente. Quali saranno gli effettivi nodi sorgente e destinazione verrà specificato nel file di dati.

A questo punto i dati relativi all'esempio li possiamo inserire nel file MAX-FLOW.DAT. In questo caso i nodi sorgente e destinazione sono, rispettivamente, i nodi S e D.

MAX-FLOW.DAT

INSIEMI

set $NODI$ **:=** S n1 n2 n3 n4 D **;**
set $ARCHI$ **:=** (S,n1) (S,n2) (n1,n3) (n1,n4) (n2,n3) (n2,n4) (n3,D) (n4,D) **;**

PARAMETRI

param Sorgente := S **;**
param Destinazione := D **;**
param d **:=**
S n1 3
S n2 3

n1 n3 1
n1 n4 4
n2 n3 1
n2 n4 1
n3 D 1
n4 D 7 **;**

2.3.3 Problema del trasporto

Supponiamo di avere m *depositi* in cui è immagazzinato un prodotto e n negozi che richiedono tale prodotto. Nel deposito i è immagazzinata la quantità a_i di prodotto. Nel negozio j si richiede la quantità b_j di prodotto. È noto che il costo di trasporto di un'unità di prodotto dal deposito i al negozio j è pari a c_{ij}. Il problema del trasporto consiste nel determinare quale quantità di prodotto inviare da ciascun deposito verso ciascun negozio in modo tale da minimizzare il costo complessivo di trasporto, rispettando i vincoli sulle quantità di prodotto presenti in ciascun deposito e quelli di richieste di ciascun negozio. Si suppone che la quantità totale immagazzinata in tutti i depositi sia pari alla quantità totale richiesta da tutti i magazzini, ovvero

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j. \tag{2.12}$$

Non si tratta comunque di un'ipotesi restrittiva dal momento che ci si può sempre ricondurre a essa. Infatti, si supponga che

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j.$$

cioè nei depositi vi sia più prodotto di quanto effettivamente richiesto dai negozi. Per soddisfare l'ipotesi (2.12) basta aggiungere un negozio fittizio $n+1$ con

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j,$$

e con $c_{i,n+1} = 0$ per ogni i, $i = 1, \ldots, m$, cioè il costo del trasporto verso il negozio fittizio è pari a 0. La quantità di prodotto che un deposito invia a un negozio fittizio resta in realtà immagazzinata nel deposito. Analogamente, si supponga che

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j.$$

cioè nei depositi vi sia meno prodotto di quanto effettivamente richiesto dai negozi. Per soddisfare l'ipotesi (2.12) basta aggiungere un deposito fittizio $m+1$ con

$$a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i,$$

e con $c_{m+1,j} = 0$ per ogni j, $j = 1, \ldots, n$, cioè il costo del trasporto dal deposito fittizio è pari a 0. La quantità di prodotto che un negozio riceve da un deposito fittizio equivale in realtà a una richiesta non soddisfatta per quel negozio.

Modello matematico del problema

Vediamo ora di formulare il modello matematico del problema del trasporto.

Variabili Ad ogni coppia deposito i-negozio j associamo una variabile x_{ij} che corrisponde alla quantità di prodotto inviata dal deposito i verso il negozio j. Tale quantità dovrà essere ovviamente non negativa, ovvero:

$$x_{ij} \geq 0 \quad i = 1, \ldots, m \;\; j = 1, \ldots, n$$

e intera se il prodotto trasportato non è frazionabile.

Vincoli Per ogni deposito i la quantità totale di prodotto inviata da esso deve essere pari alla quantità di prodotto a_i in esso immagazzinata. La quantità totale di prodotto inviata dal deposito i è data dalla formula $\sum_{j=1}^{n} x_{ij}$ e quindi per ogni deposito i avremo il seguente vincolo:

$$\sum_{j=1}^{n} x_{ij} = a_i \quad i = 1, \ldots, m.$$

Analogamente, per ogni negozio j la quantità totale di prodotto ricevuta da esso deve essere pari alla quantità di prodotto b_j da esso richiesta. La quantità totale di prodotto ricevuta dal negozio j è data dalla formula $\sum_{i=1}^{m} x_{ij}$ e quindi per ogni negozio j avremo il seguente vincolo:

$$\sum_{i=1}^{m} x_{ij} = b_j \quad j = 1, \ldots, n.$$

Obiettivo Se inviare un'unità di prodotto dal deposito i al negozio j ha costo pari a c_{ij}, inviarne una quantità x_{ij} ha costo pari a $c_{ij}x_{ij}$. Sommando su tutte le possibili coppie deposito-negozio, abbiamo la seguente formula per l'obiettivo:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Quello che desideriamo è minimizzare questo costo totale di trasporto.

Riassumendo, il modello matematico del problema del trasporto è il seguente:

$$\begin{array}{ll} \min & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ & \sum_{j=1}^{n} x_{ij} = a_i \qquad i = 1,\ldots,m \\ & \sum_{i=1}^{m} x_{ij} = b_j \qquad j = 1,\ldots,n \\ & x_{ij} \geq 0 \ \text{ interi} \qquad i = 1,\ldots,m \ j = 1,\ldots,n \end{array}$$

Possiamo anche associare al problema un grafo bipartito completo $K_{m,n}$ dove su un lato della bipartizione compaiono i nodi corrispondenti ai depositi, numerati da 1 a m, mentre sull'altro lato della bipartizione compaiono i nodi corrispondenti ai negozi, numerati da $m+1$ a $m+n$ (si veda la Figura 2.4). Vale la

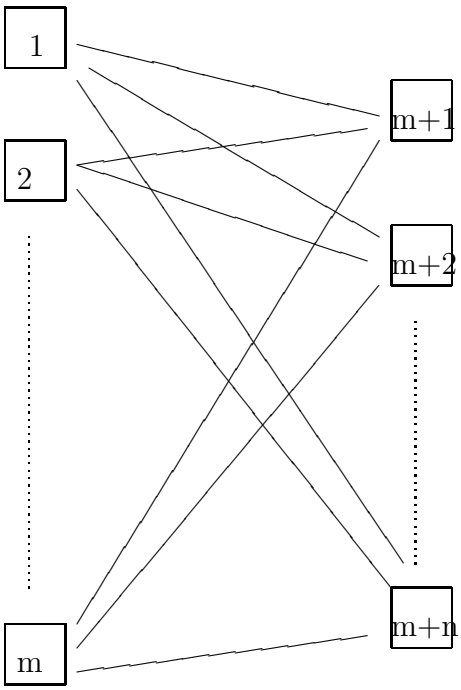


Figura 2.4: Il grafo bipartito associato a un problema del trasporto.

seguente osservazione.

Osservazione 3 *La matrice dei vincoli di uguaglianza del problema del trasporto coincide con la matrice di incidenza nodo-arco del grafo bipartito completo associato al problema.*

Esempio 9 *Si consideri il problema del trasporto con 2 depositi e 3 negozi e con*

$$a_1 = 30 \quad a_2 = 20 \quad b_1 = 15 \quad b_2 = 10 \quad b_3 = 25.$$

Tabella 2.1:

	1	2	3
1	$c_{11} = 4$	$c_{12} = 7$	$c_{13} = 5$
2	$c_{21} = 2$	$c_{22} = 4$	$c_{23} = 3$

Tabella 2.2:

	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}
Deposito 1	1	1	1	0	0	0
Deposito 2	0	0	0	1	1	1
Negozio 1	1	0	0	1	0	0
Negozio 2	0	1	0	0	1	0
Negozio 3	0	0	1	0	0	1

Per prima cosa notiamo che

$$\sum_{i=1}^{2} a_i = 30 + 20 = \sum_{j=1}^{3} b_j = 15 + 10 + 25.$$

Si supponga inoltre che i costi unitari di trasporto per le diverse coppie deposito-negozio siano date dalla Tabella 2.1 Il modello matematico di questo problema è il seguente:

$$\begin{aligned} \min \quad & 4x_{11} + 7x_{12} + 5x_{13} + 2x_{21} + 4x_{22} + 3x_{23} \\ & x_{11} + x_{12} + x_{13} = 30 \\ & x_{21} + x_{22} + x_{23} = 20 \\ & x_{11} + x_{21} = 15 \\ & x_{12} + x_{22} = 10 \\ & x_{13} + x_{23} = 25 \\ & x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0 \qquad \textit{intere} \end{aligned}$$

La matrice dei vincoli è data nella Tabella 2.2 e si può vedere che coincide con la matrice di incidenza nodo-arco del grafo bipartito completo $K_{2,3}$ associato a questo problema e illustrato in Figura 2.5.

Modello AMPL

Vediamo ora il modello AMPL per il problema del trasporto.

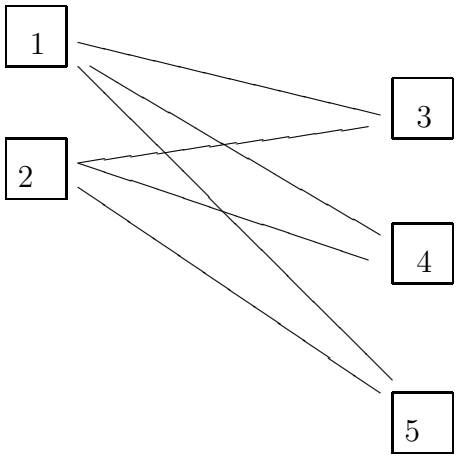


Figura 2.5: Il grafo bipartito associato al problema del trasporto dell'esempio.

TRASP.MOD

INSIEMI

set $DEPOSITI$ **;**
set $NEGOZI$ **;**

PARAMETRI

param $c\{DEPOSITI, NEGOZI\} >= 0$ **;**
param $a\{DEPOSITI\} >= 0$ **;**
param $b\{NEGOZI\} >= 0$ **;**

check : sum{i **in** $DEPOSITI$} a[i] = sum{j **in** $NEGOZI$} b[j] **;**

VARIABILI

var $x\{DEPOSITI, NEGOZI\} >= 0$, **integer ;**

VINCOLI

subject to disp_depositi {i **in** $DEPOSITI$} **:** sum{j **in** $NEGOZI$} x[i,j] =

a[i] **;**

subject to rich_negozi {j **in** $NEGOZI$} **:** sum{i **in** $DEPOSITI$} x[i,j] = b[j]
;

OBIETTIVO

minimize costo_totale **:** sum{i **in** $DEPOSITI$, j **in** $NEGOZI$} c[i,j]*x[i,j] **;**

Nel modello notiamo l'uso di **check**: la condizione dopo **:** viene controllata e se non è soddisfatta viene segnalato un errore.

I dati relativi all'esempio li possiamo inserire nel file TRASP.DAT.

TRASP.DAT

INSIEMI

set $DEPOSITI$ **:=** D1 D2 **;**
set $NEGOZI$ **:=** N1 N2 N3 **;**

PARAMETRI

param a **:=**
D1 30
D2 20 **;**

param b **:=**
N1 15
N2 10
N3 25 **;**

param c **:**

	$N1$	$N2$	$N3$	**:=**
$D1$	4	7	5	
$D2$	2	4	3	**;**

2.3.4 Problema dell'assegnamento

Siano dati due insiemi A e B entrambi di cardinalità n. Ad ogni coppia $(a_i, b_j) \in A \times B$ è associato un valore $d(a_i, b_j) \geq 0$ che misura la incompatibilità tra a_i e b_j (tanto più $d(a_i, b_j)$ è grande, quanto più a_i e b_j sono tra loro incompatibili). Il problema di assegnamento è il seguente:

Problema 1 *Individua n coppie di elementi appartenenti ad $A \times B$ in modo tale che ogni elemento di A e di B deve appartenere ad una ed una sola coppia ed in modo tale da minimizzare la incompatibilità totale, data dalla somma delle incompatibilità di ogni singola coppia.*

Nel caso $n = 3$, le seguenti tre coppie

$$(a_1, b_2) \quad (a_2, b_3) \quad (a_3, b_1)$$

rappresentano una soluzione ammissibile del problema, in quanto ogni elemento di A e B è contenuto in una ed una sola coppia. Per tale soluzione la incompatibilità totale è pari a

$$d(a_1, b_2) + d(a_2, b_3) + d(a_3, b_1).$$

Le tre coppie

$$(a_1, b_2) \quad (a_1, b_3) \quad (a_3, b_1)$$

non rappresentano invece una soluzione ammissibile in quanto, per esempio, l'elemento a_1 è presente in due coppie. Si noti che il numero di soluzioni ammissibili è pari a $n!$.
Un tipico esempio di problema di assegnamento è quello in cui si hanno n lavori da compiere (insieme A) e n lavoratori (insieme B). Dato il lavoratore b_j ed il lavoro a_i, il valore $d(a_i, b_j)$ misura l'attitudine del lavoratore b_j a compiere il lavoro a_i (tanto maggiore è tale valore, quanto minore è l'attitudine). Il problema dell'assegnamento quindi cerca di accoppiare lavori e lavoratori in modo tale da rendere minima l'incompatibilità complessiva tra questi.

Si è supposto che A e B abbiano la stessa cardinalità n. Vi sono però casi in cui questo non è vero. Nell'esempio dei lavori, vi possono essere più lavori che lavoratori ($| A |>| B |$), o più lavoratori che lavori ($| A |<| B |$). Questi casi possono **sempre** essere ricondotti al caso $| A |=| B |$ aggiungendo elementi fittizi. Per esempio, nel caso vi siano più lavoratori che lavori, si aggiungono $| B | - | A |$ lavori fittizi a_i e per ognuno di questi la sua incompatibilità $d(a_i, b_j)$ viene fissata a 0 per ogni lavoratore b_j. In questo modo ci si è ricondotti al caso con $| A |=| B |$. Assegnare un lavoro fittizio ad un lavoratore equivale a non

assegnargli alcun lavoro.

A un problema di assegnamento si associa un grafo bipartito completo $K_{n,n}$ con n nodi associati agli elementi a_i dell'insieme A su un lato della bipartizione e n nodi associati agli elementi b_j dell'insieme B sull'altro lato della bipartizione.

Esempio 10 *Si consideri l'insieme A formato dagli elementi a_1, a_2, a_3, a_4 e l'insieme B formato dagli elementi b_1, b_2, b_3, b_4. I valori di incompatibilità $d(a_i, b_j)$, $i, j = 1, 2, 3, 4$, tra elementi dell'insieme A ed elementi dell'insieme B sono riportati nella seguente matrice:*

$$\begin{bmatrix} 2 & 3 & 4 & 5 \\ 6 & 2 & 2 & 2 \\ 7 & 2 & 3 & 3 \\ 2 & 3 & 4 & 5 \end{bmatrix}, \tag{2.13}$$

mentre il grafo bipartito completo $K_{4,4}$ associato al problema è riportato in Figura 2.6.

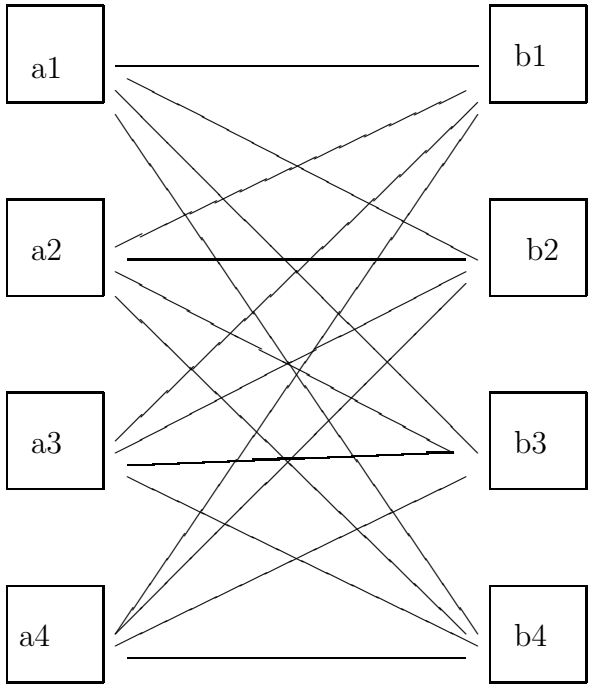


Figura 2.6: Il grafo bipartito completo $K_{4,4}$ associato al problema dell'esempio.

Modello matematico del problema

Indichiamo più semplicemente con d_{ij} i valori $d(a_i, b_j)$.

Variabili Ad ogni coppia $(a_i, b_j) \in A \times B$ si associa una variabile x_{ij} con i seguenti possibili valori:

$$x_{ij} = \begin{cases} 1 & \text{se } a_i \text{ è assegnato a } b_j \\ 0 & \text{altrimenti} \end{cases}$$

Quindi $x_{ij} \in \{0, 1\}$.

Vincoli I vincoli sono i seguenti. Ad ogni elemento a_i è assegnato uno ed un solo b_j e quindi avremo i seguenti n vincoli

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i \in \{1, \ldots, n\};$$

ad ogni elemento b_j è assegnato uno ed un solo a_i e quindi avremo i seguenti n vincoli

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall\, j \in \{1, \ldots, n\};$$

Obiettivo Il contributo all'incompatibilità totale di (a_i, b_j) è d_{ij} se a_i viene assegnato a b_j, cioè $x_{ij} = 1$, ed è 0 se a_i non viene assegnato a b_j, cioè $x_{ij} = 0$. In entrambi i casi il contributo all'incompatibilità totale di (a_i, b_j) è $d_{ij}x_{ij}$. Sommando su tutte le possibili coppie si ottiene l'obiettivo del problema:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}.$$

Quindi, riassumendo, il problema di assegnamento è un problema di PLI (meglio ancora, di PL binaria) con la seguente forma

$$\begin{array}{lll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} & \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall\, i \in \{1, \ldots, n\} \\ & \sum_{i=1}^{n} x_{ij} = 1 & \forall\, j \in \{1, \ldots, n\} \\ & x_{ij} \in \{0, 1\} & \forall\, i, j \end{array}$$

I vincoli $x_{ij} \in \{0, 1\}$ possono essere sostituiti con i vincoli $0 \leq x_{ij} \leq 1$, x_{ij} intere, ottenendo quindi il seguente problema di PLI:

$$\begin{array}{lll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} & \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall\, i \in \{1, \ldots, n\} \\ & \sum_{i=1}^{n} x_{ij} = 1 & \forall\, j \in \{1, \ldots, n\} \\ & 0 \leq x_{ij} \leq 1 \quad \text{intere} & \forall\, i, j \end{array}$$

Ma possiamo notare che:

$$x_{ij} \geq 0, \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i \in \{1, \ldots, n\} \quad \Rightarrow \quad x_{ij} \leq 1 \;\; \forall\, i, j \in \{1, \ldots, n\},$$

il che significa che i vincoli $x_{ij} \leq 1$ sono del tutto inutili e possono quindi essere eliminati. Possiamo quindi riscrivere il modello nella seguente forma:

$$\begin{array}{lll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} & \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall\, i \in \{1, \ldots, n\} \\ & \sum_{i=1}^{n} x_{ij} = 1 & \forall\, j \in \{1, \ldots, n\} \\ & x_{ij} \geq 0 \quad \text{intere} & \forall\, i, j \end{array} \tag{2.14}$$

Una volta riscritto in questo modo, notiamo che il problema di assegnamento è un caso particolare di problema del trasporto dove vi sono n depositi e n negozi, ogni deposito ha una sola unità di prodotto a disposizione e ogni negozio richiede una sola unità di prodotto.

Modello AMPL

Vediamo ora il modello AMPL per il problema di assegnamento.

ASSEGN.MOD

INSIEMI

set A **;**
set B **;**

PARAMETRI

param $d\{A, B\}$ **;**

check : card(A) = **card**(B) **;**

VARIABILI

var $x\{A, B\}$ **binary ;**

VINCOLI

subject to one_A $\{i$ **in** $A\}$ **:** sum$\{j$ **in** $B\}$ x[i,j] $= 1$ **;**

subject to one_B $\{j$ **in** $B\}$ **:** sum$\{i$ **in** $A\}$ x[i,j] $= 1$ **;**

OBIETTIVO

minimize costo_totale **:** sum$\{i$ **in** A, j **in** $B\}$ d[i,j]*x[i,j] **;**

Nel modello notiamo l'uso di **card** che, avendo come argomento un insieme, restituisce la cardinalità di tale insieme. I dati relativi all'esempio li possiamo inserire nel file ASSEGN.DAT.

ASSEGN.DAT

INSIEMI

set A **:=** a1 a2 a3 a4 **;**
set B **:=** b1 b2 b3 b4 **;**

PARAMETRI

param d **:**

	$b1$	$b2$	$b3$	$b4$	**:=**
$a1$	2	3	4	5	
$a2$	6	2	2	2	
$a3$	7	2	3	3	
$a4$	2	3	4	5	**;**

2.3.5 Problemi di flusso multi-commodity

Quando abbiamo descritto il problema di flusso a costo minimo, abbiamo dato per scontato che nella rete viaggiasse un singolo tipo di prodotto. In realtà in

molte applicazioni si ha che lungo la rete, rappresentata tramite il grafo orientato $G = (V, A)$, viaggiano contemporaneamente più tipi di prodotto (multicommodity). Supponiamo di avere $r > 1$ di questi tipi e che per ogni tipo $k \in \{1, \ldots, r\}$ di prodotto vi sia un unico nodo sorgente, indicato con S_k, con valore associato b_k e un unico nodo destinazione D_k con valore associato $-b_k$. Anche qui avremo un costo di trasporto unitario c_{ij} lungo ogni arco $(i, j) \in A$ e una capacità d_{ij} associata allo stesso arco.

Modello matematico

Vediamo ora di ricavare il modello matematico del problema di flusso multicommodity.

Variabili Rispetto al problema di flusso a costo minimo, ora le variabili avranno un indice in più relativo al tipo di prodotto:

$$x_{ij}^k = \text{quantità di prodotto } k \text{ inviata lungo l'arco } (i, j)$$

ovviamente vincolate a essere non negative e anche intere, nel caso il prodotto k non sia frazionabile.

Vincoli Come vincoli dovremo includere quelli tra differenza del flusso in uscita e in entrata, distinguendo tra i flussi dei diversi tipi di prodotti:

$$\sum_{h:(i,h)\in A} x_{ih}^k - \sum_{h:(h,i)\in A} x_{hi}^k = \begin{cases} b_k & \text{se } i \equiv S_k \\ -b_k & \text{se } i \equiv D_k \\ 0 & \text{altrimenti} \end{cases}$$

per ogni $i \in V$ e per ogni $k \in \{1, \ldots, r\}$. Avremo poi i vincoli sulla capacità degli archi:

$$\sum_{k=1}^{r} x_{ij}^k \leq d_{ij} \quad \forall\ (i, j) \in A.$$

Obiettivo L'obiettivo, con le opportune modifiche rispetto al problema di flusso a costo minimo, sarà il seguente:

$$\sum_{(i,j)\in A} [c_{ij} \sum_{k=1}^{r} x_{ij}^k]$$

dove la sommatoria più interna rappresenta la quantità totale (su tutti gli r prodotti) di merce trasportata lungo l'arco (i, j). Naturalmente l'obiettivo sarà da minimizzare. Nel caso in cui anche il costo unitario di trasporto dipenda dal tipo di prodotto k trasportato (c_{ij}^k), potremo modificare l'obiettivo in questo modo:

$$\sum_{(i,j)\in A} \sum_{k=1}^{r} c_{ij}^k x_{ij}^k.$$

Riassumendo, il modello del problema di flusso multi-commodity è il seguente:

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A}[c_{ij}\sum_{k=1}^{r} x_{ij}^k] \\
& \sum_{h:(i,h)\in A} x_{ih}^k - \sum_{h:(h,i)\in A} x_{hi}^k = \begin{cases} b_k & \text{se } i \equiv S_k \\ -b_k & \text{se } i \equiv D_k \\ 0 & \text{altrimenti} \end{cases} \quad \forall\ i \in V,\ \forall\ k \in \{1,\ldots,r\} \\
& \sum_{k=1}^{r} x_{ij}^k \leq d_{ij} \qquad \forall\ (i,j) \in A \\
& x_{ij}^k \geq 0 \quad \text{interi} \qquad \forall\ (i,j) \in A,\ \forall\ k \in \{1,\ldots,r\}.
\end{array}
$$

Modello AMPL

Vediamo ora il modello AMPL per il problema di flusso multi-commodity.

MULTI-COMMODITY.MOD

INSIEMI

set $NODI$ **;**
set $ARCHI$ **within** $NODI$ **cross** $NODI$ **;**
set $PRODOTTI$ **;**

PARAMETRI

param Sorgente{$PRODOTTI$} **symbolic in** $NODI$ **;**
param Destinazione{k **in** $PRODOTTI$} **symbolic in** $NODI$, **!=** Sorgente[k]
;
param b{$PRODOTTI$} >= 0 **;**
param c{$ARCHI$} **;**
param d{$ARCHI$} >= 0, **default** Infinity **;**

VARIABILI

var x{$ARCHI$, $PRODOTTI$} >= 0, **integer ;**

VINCOLI

subject to equilibrio{j **in** $NODI$, k **in** $PRODOTTI$ **:** j **!=** Sorgente[k] **and**
j **!=** Destinazione[k] } **:** sum{h **in** $NODI$: (j,h) **in** $ARCHI$} x[j,h,k] - sum{h
in $NODI$: (h,j) **in** $ARCHI$} x[h,j,k] = 0 **;**

subject to bilancio_sorgente{k **in** $PRODOTTI$ } **:** sum{h **in** $NODI$: (Sorgente[k],h) **in** $ARCHI$} x[Sorgente[k],h,k] - sum{h **in** $NODI$: (h,Sorgente[k]) **in** $ARCHI$} x[h,Sorgente[k],k] = b[k] **;**

subject to bilancio_destinazione{k **in** $PRODOTTI$ } **:** sum{h **in** $NODI$: (Destinazione[k],h) **in** $ARCHI$} x[Destinazione[k],h,k] - sum{h **in** $NODI$: (h,Destinazione[k]) **in** $ARCHI$} x[h,Destinazione[k],k] = -b[k] **;**

subject to cap_max{(i,j) **in** $ARCHI$ } **:** sum{k **in** $PRODOTTI$} x[i,j,k] <= d[i,j] **;**

OBIETTIVO

minimize costo_totale **:** sum{(i,j) **in** $ARCHI$} c[i,j]*(sum{k **in** $PRODOTTI$} x[i,j,k]) **;**

2.3.6 Problema del commesso viaggiatore

In una determinata zona abbiamo n centri con distanze d_{ij} tra ogni coppia di centri i e j, $i \neq j$. Un commesso viaggiatore deve partire da uno di questi centri (supporremo nel seguito sia il centro 1) e visitare gli altri $n-1$ prima di tornare a quello di partenza. Quello che vorrebbe il commesso viaggiatore è individuare la sequenza di visita dei centri che renda minima la distanza che complessivamente percorre. Il problema verrà indicato più avanti anche come problema `TSP` (Travelling Salesman Problem). Anche qui i grafi ci aiutano a rappresentare la situazione. Costruiamo un grafo completo i cui nodi corrispondano agli n centri. Gli archi indicano i collegamenti tra i centri e potranno essere non orientati se le distanze tra centri sono simmetriche, ovvero per ogni i, j si ha $d_{ij} = d_{ji}$, oppure orientati se non c'è simmetria nelle distanze (si pensi al caso di strade percorribili in un solo senso che rendono la distanza tra due centri diversa a seconda del verso di percorrenza). Il problema sul grafo consiste nell'individuare tra tutti i circuiti hamiltoniani (si veda la Definizione 20) quello con distanza complessiva minima. Come al solito gli ambiti applicativi di un problema di questo tipo sono molto diversi tra loro. Si pensi al sequenziamento di una serie di n operazioni che devono essere eseguite su una certa macchina. Il passaggio tra due operazioni eseguite una dietro l'altra non avviene in modo istantaneo ma richiede un certo tempo di setup, indicato con d_{ij}, per passare dalla configurazione della macchina per eseguire l'operazione i a quella per es-

eguire l'operazione j. Ciò che si desidera è stabilire la sequenza in cui eseguire le n operazioni (ipotizzando che queste vengano svolte ciclicamente e che quindi una volta raggiunta l'ultima si debba ripartire dalla prima) in modo tale da minimizzare i tempi totali di setup.

Modello matematico

Associamo ad ogni arco $(i, j) \in A$ la variabile x_{ij} che potrà assumere i seguenti valori:

$$x_{ij} = \begin{cases} 1 & \text{se l'arco } (i,j) \text{ fa parte del circuito hamiltoniano} \\ 0 & \text{altrimenti} \end{cases}$$

Quindi $x_{ij} \in \{0, 1\}$, ovvero è binaria. Vediamo ora quali vincoli devono essere soddisfatti dai circuiti hamiltoniani. Prendiamo in esame la Figura 2.7 dove gli archi tratteggiati rappresentano il circuito hamiltoniano

$$1 \to 3 \to 2 \to 4 \to 1.$$

sul grafo completo con insieme di nodi $V = \{1, 2, 3, 4\}$. Si nota che in ogni nodo

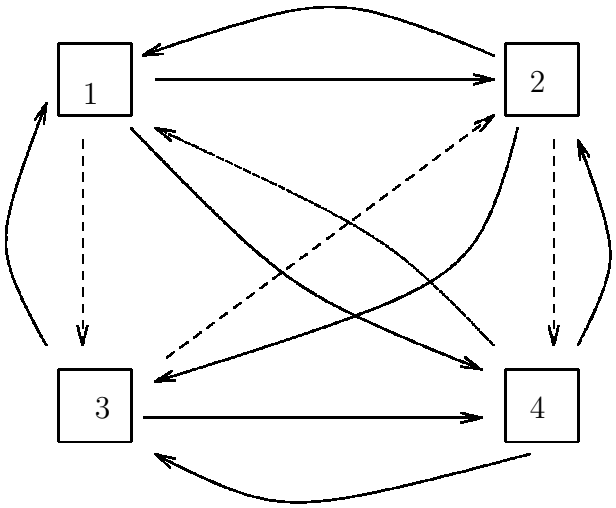


Figura 2.7: Un circuito hamiltoniano rappresentato dagli archi tratteggiati.

c'è esattamente un arco del circuito che entra nel nodo ed esattamente uno che esce dal nodo. Avremo quindi i seguenti due insiemi di vincoli:

1. Per ogni nodo $j \in V$:
$$\sum_{i \in V,\ i \neq j} x_{ij} = 1,$$
ovvero il circuito può contenere uno ed un solo arco entrante in j.

2. Per ogni nodo $j \in V$:

$$\sum_{i \in V,\ i \neq j} x_{ji} = 1,$$

ovvero il circuito può contenere uno ed un solo arco uscente da j.

Nel nostro esempio con 4 vertici avremo i seguenti 8 vincoli

$$x_{12} + x_{13} + x_{14} = 1 \quad x_{21} + x_{23} + x_{24} = 1$$

$$x_{31} + x_{32} + x_{34} = 1 \quad x_{41} + x_{42} + x_{43} = 1$$

$$x_{21} + x_{31} + x_{41} = 1 \quad x_{12} + x_{32} + x_{42} = 1$$

$$x_{13} + x_{23} + x_{43} = 1 \quad x_{14} + x_{24} + x_{34} = 1.$$

Tutti i circuiti hamiltoniani soddisfano questi vincoli. Ma è vero anche il viceversa? Cioè è vero che tutte le soluzioni che soddisfano tali vincoli rappresentano circuiti hamiltoniani? La risposta è no e lo si mostra attraverso il nostro esempio. Prendiamo la soluzione

$$x_{12} = x_{21} = x_{34} = x_{43} = 1$$
$$x_{13} = x_{31} = x_{14} = x_{41} = x_{23} = x_{32} = x_{24} = x_{42} = 0.$$

Questa non rappresenta un circuito hamiltoniano come si vede dalla Figura 2.8, ma si può verificare che soddisfa i vincoli introdotti (in ogni nodo entra ed esce un solo arco). Ciò si verifica per la presenza di sottocircuiti (nell'esempio

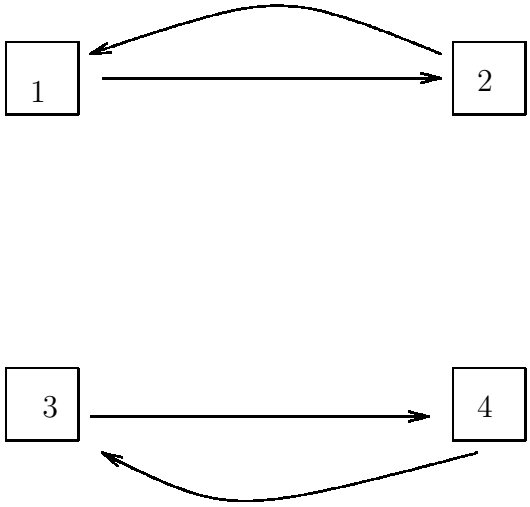


Figura 2.8: Una soluzione non ammissibile rappresentata da due sottocircuiti.

$1 \to 2 \to 1$ e $3 \to 4 \to 3$). Per avere un insieme di vincoli che rappresenta tutti e soli i circuiti hamiltoniani dobbiamo aggiungere ai vincoli già introdotti

altri vincoli. Vedremo due gruppi di vincoli che consentono di scartare solo i sottocircuiti, senza essere violati dai circuiti hamiltoniani.

Nella prima classe di vincoli, associamo a ogni nodo i diverso dal nodo di partenza ($i \neq 1$) una variabile y_i a valori interi compresi tra 1 e $n-1$. Questa variabile indica la posizione del nodo i nella sequenza di visita dei nodi. Per il nodo di partenza 1 tale posizione è fissata a 0, ovvero porremo sempre $y_1 = 0$. Se nel circuito è presente l'arco (i, j), il nodo j segue il nodo i nel ciclo e quindi la posizione di j deve essere di almeno un'unità superiore a quella di i, ovvero avremo il seguente vincolo logico:

$$x_{ij} = 1 \quad \Rightarrow \quad y_j \geq y_i + 1 \quad \forall\, i \neq j, \;\; j \neq 1. \tag{2.15}$$

A questo punto i sottocircuiti che non partono dal nodo 1 sono esclusi. Se pensiamo alla soluzione con $x_{34} = x_{43} = 1$ e quindi con il sottocircuito (che non parte dal nodo 1)

$$3 \to 4 \to 3,$$

dai vincoli (2.15) dobbiamo avere $y_4 \geq y_3 + 1$ e $y_3 \geq y_4 + 1$ che non possono essere soddisfatti contemporaneamente. Notiamo che i vincoli (2.15) *non* vengono imposti per $j = 1$ (altrimenti questi vincoli ci farebbero escludere non solo i sottocircuiti che non partono dal nodo 1, ma anche i circuiti hamiltoniani che partono dal nodo 1).

Per imporre (2.15), tenuto conto che abbiamo anche i vincoli

$$1 \leq y_i \leq n-1 \quad i = 2, \ldots, n, \quad y_1 = 0 \tag{2.16}$$

possiamo utilizzare vincoli di questo tipo:

$$y_j - y_i \geq x_{ij} + (1-n)(1-x_{ij}).$$

Infatti se $x_{ij} = 1$ abbiamo esattamente il vincolo richiesto nell'implicazione (2.15), mentre per $x_{ij} = 0$ abbiamo il vincolo

$$y_j - y_i \geq 1 - n$$

ridondante alla luce dei limiti (2.16).
Arriviamo quindi infine al seguente modello del problema:

$$\begin{array}{lcl}
\min & \sum_{i \neq j} d_{ij} x_{ij} & \\
& \sum_{i \in V,\ i \neq j} x_{ij} = 1 & \forall\ j \in V \\
& \sum_{i \in V,\ i \neq j} x_{ji} = 1 & \forall\ j \in V \\
& y_j - y_i \geq x_{ij} + (1-n)(1-x_{ij}) & \forall\ i \neq j,\ j \neq 1 \\
& x_{ij} \in \{0,1\} & \forall\ i, j \in \{1, \ldots, n\},\ i \neq j \\
& 1 \leq y_i \leq n-1 & i = 2, \ldots, n \\
& y_1 = 0 &
\end{array}$$

I vincoli (2.15) possono essere riadattati in modo tale da tener conto di altre richieste nel problema. Per esempio, supponiamo che a un arco (i, j) sia associato, oltre alla distanza d_{ij}, anche un tempo di percorrenza T_{ij} e che esistano per ogni nodo $i \neq 1$ delle finestre temporali $[Am_i, AM_i]$ all'interno delle quali i nodi devono essere visitati. In tal caso possiamo definire y_i non come *posizione del nodo i nella sequenza* ma come *istante in cui il nodo i viene visitato.* Allora il vincolo logico (2.15) viene modificato nel modo seguente

$$x_{ij} = 1 \quad \Rightarrow \quad y_j - y_i \geq T_{ij} \quad \forall\, i \neq j, \quad j \neq 1$$

esprimibile tramite la seguente disequazione

$$y_j - y_i \geq T_{ij} x_{ij} - (T_{\max} - T_{\min})(1 - x_{ij}),$$

dove, ad esempio, si può porre

$$T_{\min} = \min_{i \neq j} T_{ij} \qquad T_{\max} = n \max_{i \neq j} T_{ij}.$$

Ora, per imporre la visita del nodo i nella finestra temporale data, è sufficiente introdurre il vincolo

$$Am_i \leq y_i \leq AM_i \quad i = 2, \ldots, n.$$

Ma come si diceva, l'eliminazione dei sottocircuiti può avvenire anche tramite un'altra classe di vincoli, senza dover introdurre le nuove variabili y_i. I vincoli alternativi per l'eliminazione di tutti i sottocircuiti sono i seguenti:

$$\forall\, U \subseteq V :\ 2 \leq \mid U \mid \leq \mid V \mid -2 \qquad \sum_{i \in U,\ j \in V \setminus U} x_{ij} \geq 1. \tag{2.17}$$

Tali vincoli richiedono che per ogni possibile partizione di V in due sottinsiemi (ciascuno di cardinalità almeno pari a 2), deve esserci almeno un arco che va da un sottinsieme all'altro. Questo esclude i sottocircuiti come si può constatare dal nostro esempio: tra $U = \{1, 2\}$ e $V \setminus U = \{3, 4\}$ non si ha alcun arco. Vediamo ora di scrivere tali vincoli per il nostro esempio. Notiamo innanzitutto che, essendo $\mid V \mid = 4$, si avrà che gli insiemi U da considerare sono i sottinsiemi con cardinalità che soddisfa $2 \leq \mid U \mid \leq 4 - 2 = 2$ e quindi sono i soli sottinsiemi con cardinalità 2. Ognuno di questi darà origine ad un vincolo e quindi ai vincoli già introdotti dovremo aggiungere i seguenti:

$$\begin{aligned}
U = \{1, 2\} \ &\rightarrow \quad x_{13} + x_{14} + x_{23} + x_{24} \geq 1 \\
U = \{1, 3\} \ &\rightarrow \quad x_{12} + x_{14} + x_{32} + x_{34} \geq 1 \\
U = \{1, 4\} \ &\rightarrow \quad x_{12} + x_{13} + x_{42} + x_{43} \geq 1 \\
U = \{2, 3\} \ &\rightarrow \quad x_{21} + x_{24} + x_{31} + x_{34} \geq 1
\end{aligned}$$

$$U = \{2,4\} \;\rightarrow\; x_{21} + x_{23} + x_{41} + x_{43} \geq 1$$
$$U = \{3,4\} \;\rightarrow\; x_{31} + x_{32} + x_{41} + x_{42} \geq 1$$

Un'ulteriore semplificazione si ha osservando che dato un vincolo per il sottinsieme U, questo rende ridondante quello per il suo complementare $V \backslash U$. Quindi, possiamo restringere la nostra attenzione agli U tali che $\mid U \mid \leq \lceil \mid V \mid /2 \rceil$. Vediamo ancora come la soluzione con sottocircuiti del nostro esempio è scartata in quanto:

$$x_{13} + x_{14} + x_{23} + x_{24} = 0 < 1,$$

e quindi abbiamo un vincolo violato.

Con questi vincoli il modello matematico del problema è il seguente:

$$\begin{array}{lll} \min & \sum_{i \neq j} d_{ij} x_{ij} & \\ & \sum_{i \in V,\ i \neq j} x_{ij} = 1 & \forall\ j \in V \\ & \sum_{i \in V,\ i \neq j} x_{ji} = 1 & \forall\ j \in V \\ & \sum_{i \in U,\ j \in V \backslash U} x_{ij} \geq 1 & \forall\ U \subseteq V :\ 2 \leq \mid U \mid \leq \mid V \mid -2 \\ & x_{ij} \in \{0,1\} & \forall\ i,j \in \{1,\ldots,n\},\ i \neq j \end{array}$$

Da un lato quest'ultimo modello contiene un numero di vincoli estremamente elevato. Infatti il numero di vincoli (2.17) è $O(2^{|V|})$ e dunque esponenziale, mentre il numero di vincoli (2.15) è $O(\mid V \mid^2)$. D'altro lato però si può anche vedere che, in un senso che potremo chiarire solo più avanti, il modello basato su (2.17) è molto più preciso di quello basato su (2.15).

Modello AMPL

Vedremo ora il modello AMPL per il problema TSP con i vincoli (2.15) per l'eliminazione dei sottocircuiti.

TSP.MOD

INSIEMI

set $NODI$ **ordered ;**
set $ARCHI$ **within** ($NODI$ **cross** $NODI$) **;**

PARAMETRI

param n **:= card**($NODI$) **;**

param $d\{ARCHI\}$ **default** 100000000 **;**

VARIABILI

var $x\{ARCHI\}$ **binary ;**
var $y\{NODI\}$ >= 0, <= n-1, **integer ;**

VINCOLI

subject to ingresso$\{i$ **in** $NODI\}$ **:** sum$\{j$ **in** $NODI$: j != $i\}$ x[j,i] = 1 **;**
subject to uscita$\{i$ **in** $NODI\}$ **:** sum$\{j$ **in** $NODI$: j != $i\}$ x[i,j] = 1 **;**
subject to sequenza$\{(i,j)$ **in** $ARCHI$: j **!= first**$(NODI)\}$ **:** y[j]-y[i] >= n*x[i,j]+1-n **;**
subject to nodo_partenza **:** y[**first**$(NODI)$]=0 **;**

OBIETTIVO

minimize distanza_totale **:** sum$\{(i,j)$ **in** $ARCHI\}$ d[i,j]*x[i,j] **;**

Facciamo alcune osservazioni sul modello. L'insieme $NODI$ viene dichiarato **ordered**. L'ordine sarà definito da quello con cui vengono specificati i nodi nel file .DAT. In questo modo nel modello possiamo fare riferimento al primo elemento dell'insieme (nel nostro caso il nodo di partenza del circuito) che è semplicemente identificato da **first**$(NODI)$.
Notiamo la dichiarazione dell'insieme $ARCHI$ che coincide con il prodotto cartesiano dell'insieme $NODI$ per se stesso.
Il default 100000000 per il valore d degli archi va inteso come valore abbastanza grande da escludere a priori la presenza dell'arco in una soluzione ottima del problema.

2.3.7 Problemi di instradamento di veicoli

Alcuni problemi sono strettamente collegati a quello del commesso viaggiatore ma richiedono alcune modifiche rispetto a esso. Pensiamo al caso in cui si abbia a disposizione non un solo commesso viaggiatore ma fino a un numero massimo N di commessi viaggiatori (o, se pensiamo al sequenziamento di operazioni, non una singola macchina ma fino a N macchine uguali). Supporremo che per tutti i commessi viaggiatori il nodo di partenza sia il nodo 1. Anche qui possiamo usare le variabili binarie x_{ij} e intere y_i (con $y_1 = 0$ e le altre incluse tra 1 e $n-1$) già viste per il commesso viaggiatore. Potremo ancora utilizzare i vincoli

(2.15). Per i nodi diversi dal nodo di partenza 1 avremo sempre la richiesta di un solo arco entrante e uno solo uscente:

$$\begin{cases} \sum_{i \neq j} x_{ij} = 1 \\ \\ \sum_{i \neq j} x_{ji} = 1 \end{cases} \quad \forall\ j \neq 1$$

L'unica differenza rispetto al problema del commesso viaggiatore è che non avremo i vincoli appena visti di un solo arco entrante e uscente per quel che riguarda il nodo di partenza 1. Per tale nodo avremo la richiesta che il numero di archi complessivamente entranti deve essere pari a quello degli uscenti:

$$\sum_{i \neq 1} x_{i1} = \sum_{i \neq 1} x_{1i}$$

(ovvero il numero di commessi viaggiatori che ritornano nel nodo 1 deve essere esattamente pari a quello dei viaggiatori che sono partiti da tale nodo) ed entrambe queste quantità dovranno essere non superiori a N:

$$\sum_{i \neq 1} x_{i1} \leq N$$

(ovvero non possono uscire ed entrare dal nodo 1 più di N commessi viaggiatori). L'obiettivo non cambia rispetto al commesso viaggiatore tradizionale.

Una variante interessante di questo problema è quando abbiamo dei vincoli di capacità (di merce trasportabile) per un commesso viaggiatore (si pensi al caso in cui il commesso viaggiatore utilizzi un veicolo a capacità limitata per trasportare le merci). Supporremo che in ogni centro $i \neq 1$ il commesso viaggiatore carichi una quantità Car_i di merce. A questo punto possiamo definire le variabili y_i in questo modo:

$$y_i = \text{quantità di merce presente nel veicolo al nodo } i.$$

Su tali variabili imporremo i vincoli

$$0 \leq y_i \leq C,$$

dove C è la capacità massima di un veicolo. Avremo quindi i vincoli logici

$$x_{ij} = 1 \quad \Rightarrow \quad y_j - y_i \geq Car_j$$

esprimibili con

$$y_j - y_i \geq Car_j x_{ij} - C(1 - x_{ij}) \tag{2.18}$$

Come i vincoli (2.15) per il commesso viaggiatore originario, i vincoli (2.18) per eliminare soluzioni con sottocircuiti indesiderati (cioè sottocircuiti che non

partono dal nodo di partenza 1) non sono molto efficaci dal punto di vista algoritmico. Anche qui un'alternativa è usare un numero esponenziale di vincoli come in (2.17). Tali vincoli sono:

$$\sum_{i \notin S,\ j \in S} x_{ij} \geq \left\lceil \frac{\sum_{j \in S} Car_j}{C} \right\rceil \quad \forall\ S \subseteq V \setminus \{1\}.$$

Essi impongono che per ogni sottinsieme S di nodi non contenente il nodo di partenza 1, il numero di veicoli che dovranno entrare nei nodi in S deve essere almeno pari a:

$$\left\lceil \frac{\sum_{j \in S} Car_j}{C} \right\rceil.$$

Inoltre, tali vincoli garantiscono che i limiti di capacità dei veicoli siano rispettati (il numero di veicoli che entra in S è in grado di portare tutto il carico dei nodi in S).

Capitolo 3

La teoria della Programmazione Lineare

In questo capitolo studieremo alcuni risultati teorici sulla Programmazione Lineare, culminanti nel teorema fondamentale della stessa PL. Tali risultati porranno le fondamenta per l'algoritmo di risoluzione per i problemi di PL che verrà presentato nel capitolo successivo, l'*algoritmo del simplesso*. Verranno qui introdotti anche alcuni concetti importanti (come quelli di *base* e *soluzione di base*) che verranno poi utilizzati nella definizione dell'algoritmo.

3.1 I problemi di PL in forma canonica

Come già visto, la forma generica dei problemi di PL è la seguente

$$\begin{array}{ll} \max \text{ (o } \min) & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i \in I_1 \\ & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i \in I_2 \\ & \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i \in I_3 \\ & x_1, \ldots, x_n \in R \end{array}$$

Qui però cominceremo a concentrare l'attenzione su un tipo particolare di problemi di PL, rappresentato dai problemi di PL in *forma canonica*, in cui l'obiettivo è sempre da massimizzare, i vincoli sono tutti di $\leq$ e le variabili sono tutte vincolate ad assumere valori non negativi, cioè

$$\begin{array}{ll} \max & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \\ & x_j \geq 0 \qquad\quad j = 1, \ldots, n \end{array}$$

Come vedremo si tratta solo di una restrizione apparente rispetto ai problemi di PL in forma generica. Possiamo anche scrivere il problema di PL in forma canonica in forma più compatta introducendo dei vettori. Indichiamo con:

- $c \in R^n$ il vettore di dimensione n con componenti c_j, $j = 1, \ldots, n$, ovvero:
$$c = (c_1 \quad c_2 \quad \cdots \quad c_n);$$
- $x \in R^n$ il vettore di variabili di dimensione n con componenti x_j, $j = 1, \ldots, n$, ovvero:
$$x = (x_1 \quad x_2 \quad \cdots \quad x_n);$$
- $a_i \in R^n$, $i = 1, \ldots, m$, gli m vettori di dimensione n con componenti a_{ij}, $j = 1, \ldots, n$, ovvero:
$$a_i = (a_{i1} \quad a_{i2} \quad \cdots \quad a_{in}).$$

Recuperando alcune nozioni di algebra lineare, ricordiamo che dati due vettori della stessa dimensione n, indicati con $p = (p_1 \quad \cdots \quad p_n)$ e $q = (q_1 \quad \cdots \quad q_n)$, il prodotto scalare tra questi vettori è definito come somma dei prodotti delle singole componenti, ovvero:
$$pq = \sum_{j=1}^{n} p_j q_j.$$
Un'importante proprietà del prodotto scalare è la seguente. Siano $p, q_1, q_2 \in R^n$ e $\alpha, \beta \in R$. Allora:
$$p(\alpha q_1 + \beta q_2) = \alpha(pq_1) + \beta(pq_2)$$
Possiamo anche generalizzare questa proprietà. Infatti, dati i vettori $p, q_1, q_2, \ldots, q_t \in R^n$ e gli scalari $\alpha_1, \alpha_2, \ldots, \alpha_t \in R$, si ha che
$$p[\alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_t q_t] = p\left[\sum_{i=1}^{t} \alpha_i q_i\right] = \sum_{i=1}^{t} \alpha_i (pq_i).$$
Si ricordano infine le definizioni di prodotto di matrice per vettore e di vettore per matrice. Data una matrice A di ordine $m \times n$ (m righe e n colonne)
$$\begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix}$$
ed un vettore p di dimensione n
$$p = (p_1 \quad \cdots \quad p_n)$$
Il prodotto matrice-vettore è un vettore di dimensione m la cui componente i è il prodotto scalare tra la i-esima riga di A e il vettore p:
$$\sum_{j=1}^{n} a_{ij} p_j$$

Data una matrice A di ordine $m \times n$ (m righe e n colonne)

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$$

ed un vettore q di dimensione m

$$q = (q_1 \quad \cdots \quad q_m)$$

Il prodotto vettore-matrice è un vettore di dimensione n la cui componente j è il prodotto scalare tra la j-esima colonna di A e il vettore q:

$$\sum_{i=1}^{m} a_{ij} q_i$$

Tenuto conto di tutto questo possiamo riscrivere il problema di PL in forma canonica nella seguente forma:

$$\begin{aligned} \max \quad & cx \\ & a_i x \leq b_i \quad i = 1, \dots, m \\ & x \geq 0 \end{aligned}$$

Possiamo ulteriormente compattare la rappresentazione con l'introduzione della matrice $A \in R^{m \times n}$ che ha tante righe quanti sono i vincoli del problema (m) e la cui i-esima riga è il vettore a_i e del vettore $b = (b_1 \quad \cdots \quad b_m) \in R^m$ di dimensione m con componenti b_i, $i = 1, \dots, m$. Con l'introduzione di questi possiamo riscrivere il problema di PL in forma canonica nel seguente modo:

$$\begin{aligned} \max \quad & cx \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Come già anticipato, se apparentemente i problemi di PL in forma canonica rappresentano un sottinsieme dei problemi di PL, è in realtà possibile dimostrare che *ogni* problema di PL ha un problema di PL in forma canonica a esso equivalente, come dimostra la seguente osservazione.

Osservazione 4 *Dato un problema di PL, esiste un problema di PL in forma canonica a esso equivalente.*

Dimostrazione Si può notare che

1. ogni problema di minimo può essere trasformato in un problema di massimo sfruttando la seguente relazione

$$\min cx = -\max -cx$$

2. ogni vincolo di $\geq$ può essere trasformato in un vincolo di $\leq$ nel modo seguente
$$a_i^T x \geq b_i \quad \Rightarrow \quad -a_i x \leq -b_i$$

3. ogni vincolo di $=$ può essere trasformato in due vincoli di $\leq$ nel modo seguente
$$a_i x = b_i \quad \Rightarrow \quad a_i x \leq b_i, \ -a_i x \leq -b_i$$

4. se abbiamo una variabile $x_j \leq 0$ possiamo sostituirla nei vincoli e nell'obiettivo con la variabile
$$x'_j = -x_j \geq 0$$

5. se abbiamo una variabile x_j libera in segno, possiamo sostituirla nei vincoli e nell'obiettivo con una differenza di variabili non negative
$$x_j = x'_j - x''_j \quad x'_j, x''_j \geq 0$$

Come esercizio, si trasformi il seguente problema di PL in un problema di PL in forma canonica

$$\begin{array}{ll} \min & x_1 + x_2 + x_3 \\ & x_1 + 2x_2 - x_3 \leq 3 \\ & x_1 + 4x_2 + 5x_3 = 5 \\ & x_1 - 2x_2 + x_3 \geq 3 \\ & x_1 \geq 0 \\ & x_2 \leq 0 \\ & x_3 \text{ libera in segno} \end{array}$$

Il risultato appena citato ci consente di concentrare la nostra attenzione sui soli problemi di PL in forma canonica.

3.2 La regione ammissibile S_a

La regione ammissibile di un problema di PL in forma canonica è definita nel modo seguente:

$$S_a = \{x \in R^n : \ a_i x \leq b_i, \ i = 1, \ldots, m, \ \ x \geq 0\}.$$

Dobbiamo ora introdurre alcune definizioni.

Definizione 1 *Un insieme C si dice* convesso *se*

$$\forall \ x_1, x_2 \in C \ \forall \ \lambda \in [0,1] : \quad \lambda x_1 + (1-\lambda) x_2 \in C,$$

ovvero se dati due punti qualsiasi in C, il segmento che li congiunge è anch'esso completamente contenuto in C.

Definizione 2 *Un insieme C si dice* limitato *se esiste un $R > 0$ tale che*

$$\forall\ x \in C: \quad \|x\| \leq R,$$

dove $\|x\|$ indica la norma euclidea di x. In altre parole l'insieme C è contenuto in una sfera di raggio finito R.

Definizione 3 *Un insieme C si dice chiuso se contiene la sua frontiera.*

Definizione 4 *Si definisce* semispazio *in R^n l'insieme di punti che soddisfa una disequazione lineare in R^n:*

$$\sum_{j=1}^{n} w_j x_j \leq v$$

(in forma vettoriale: $wx \leq v$). Si definisce iperpiano *in R^n l'insieme di punti che soddisfa un'equazione lineare in R^n:*

$$\sum_{j=1}^{n} w_j x_j = v$$

(in forma vettoriale: $wx = v$).

Definizione 5 *Si definisce* poliedro *l'intersezione di un numero finito di semispazi e/o iperpiani. Se il poliedro è limitato esso viene chiamato* politopo.

Questo ci dice che la regione ammissibile S_a di un problema di PL è un poliedro. Si noti che ogni iperpiano e ogni semispazio sono insiemi chiusi. Poiché un intersezione di un numero finito di insiemi chiusi è un insieme chiuso, ogni poliedro è un insieme chiuso. In problemi con 2 sole variabili è possibile rappresentare graficamente le regioni ammissibili S_a. Nel seguente esempio, la rappresentazione grafica ci consentirà di visualizzare le diverse forme possibili di S_a.

Esempio 11 *Rappresentare graficamente le regioni ammissibili S_a per i seguenti tre problemi e constatare che sono altrettanti esempi di regione ammissibile vuota, di politopo e di poliedro illimitato.*

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 \leq -1 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 - x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

La seguente osservazione ci mostra anche che la regione ammissibile S_a di un problema in forma canonica è un insieme convesso (il risultato è facilmente estendibile a ogni poliedro).

Osservazione 5 *La regione ammissibile S_a di un problema di PL in forma canonica è un insieme convesso.*

Dimostrazione Siano dati due generici punti $x_1, x_2 \in S_a$. Si avrà:

$$a_i x_1 \leq b_i \quad i = 1, \ldots, m \quad x_1 \geq 0,$$

e

$$a_i x_2 \leq b_i \quad i = 1, \ldots, m \quad x_2 \geq 0.$$

Quindi, per ogni $\lambda \in (0,1)$ e per ogni $i \in \{1, \ldots, m\}$ avremo:

$$a_i[\lambda x_1 + (1-\lambda)x_2] = \lambda a_i x_1 + (1-\lambda) a_i x_2 \leq \lambda b_i + (1-\lambda) b_i = b_i,$$

e

$$\underbrace{\lambda}_{>0}\underbrace{x_1}_{\geq 0} + \underbrace{(1-\lambda)}_{>0}\underbrace{x_2}_{\geq 0} \geq 0,$$

da cui

$$\lambda x_1 + (1-\lambda)x_2 \in S_a.$$

Ciò equivale a dire che S_a è un insieme convesso.

Vediamo ora di introdurre la definizione di alcuni particolari punti di S_a, i vertici di S_a.

Definizione 6 *Si definisce* vertice *di S_a un punto $\overline{x} \in S_a$ tale che non esistono due punti distinti $x_1, x_2 \in S_a$, $x_1 \neq x_2$, tali che*

$$\overline{x} = \frac{1}{2}x_1 + \frac{1}{2}x_2.$$

In problemi con 2 variabili i vertici sono facilmente identificabili, coincidendo con la usuale definizione di vertice di una figura piana.

Esempio 12 *Si verifichi che, dato il problema*

$$\begin{array}{ll} \max & x_1 + x_2 \\ & x_1 + x_2 \leq 1 \\ & -2x_1 - 2x_2 \leq -2 \\ & x_1, x_2 \geq 0 \end{array}$$

il punto $(1/2, 1/2)$ non è un vertice di S_a mentre lo è il punto $(1, 0)$

Un primo importante teorema per la PL è il seguente.

Teorema 1 *Dato un problema di PL in forma canonica, se $S_a \neq \emptyset$, allora S_a contiene almeno un vertice.*

Si può inoltre dimostrare la seguente osservazione.

Osservazione 6 *S_a ha sempre un numero finito di vertici.*

Nel caso S_a sia un poliedro illimitato possiamo anche introdurre le definizioni di raggio e raggio estremo.

Definizione 7 *Si definisce* raggio *di S_a un vettore $r \neq 0$ tale che*

$$\forall\ x_0 \in S_a \quad \forall\ \lambda \geq 0 : \quad x_0 + \lambda r \in S_a,$$

cioè la semiretta con origine in x_0 e direzione r è completamente contenuta in S_a per qualsiasi punto $x_0 \in S_a$. Un raggio r di S_a si definisce raggio estremo *di S_a se non esistono altri due raggi r_1 e r_2 di S_a con direzioni distinte, ovvero*

$$r_1 \neq \mu r_2 \quad \forall\ \mu \in R,$$

tali che

$$r = \frac{1}{2} r_1 + \frac{1}{2} r_2.$$

Ovviamente i politopi non hanno alcun raggio. Infatti l'esistenza di un raggio implica che S_a contenga almeno una semiretta, che è un insieme illimitato, mentre i politopi sono, per definizione, insiemi limitati. Anche in questo caso in problemi con 2 sole variabili è facile riconoscere i raggi estremi, che coincidono con le semirette che delimitano la figura piana.

Esempio 13 *Si verifichi che dato il problema*

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 - x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

i vettori

$$(1/2, 1) \quad (0, 1) \quad (1, 1)$$

sono tutti raggi di S_a ma solo gli ultimi due sono raggi estremi.

Come per i vertici, anche per i raggi estremi si può dimostrare che il loro numero è sempre finito.

Osservazione 7 *S_a ha sempre un numero finito di raggi estremi.*

Siamo ora pronti per enunciare un importante teorema che mostra che la regione ammissibile S_a di un problema di PL in forma canonica è completamente caratterizzata dai suoi vertici e raggi estremi , il *teorema di rapprsentazione* per S_a.

Teorema 2 *Sia dato un problema di PL in forma canonica con $S_a \neq \emptyset$. Siano $v_1, \ldots, v_k$ i vertici di S_a e, nel caso in cui S_a sia un poliedro illimitato, siano $r_1, \ldots, r_h$ i raggi estremi di S_a. Allora*

$$x \in S_a$$

se e solo se

$$\exists\ \lambda_1, \ldots, \lambda_k \geq 0, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \exists\ \mu_1, \ldots, \mu_h \geq 0$$

tali che

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{h} \mu_j r_j.$$

Il teorema ci dice che *tutti e soli* i punti di S_a sono ottenibili come somma di una combinazione convessa (combinazione lineare con coefficienti non negativi e la cui somma è pari a 1) dei vertici di S_a e di una combinazione lineare con coefficienti non negativi dei raggi estremi di S_a.

3.3 L'insieme delle soluzioni ottime S_{ott}

Fino a questo momento ci siamo limitati a considerare la regione ammissibile S_a di un problema di PL. Ricordiamo però che il nostro scopo è determinare una soluzione ottima del problema di PL. Quindi dobbiamo trovare almeno un punto all'interno dell'insieme:

$$S_{ott} = \{x^* \in S_a :\ cx^* \geq cx \ \ \forall\ x \in S_a\},$$

detto insieme delle soluzioni ottime del problema. Notiamo immediatamente che $S_{ott} \subseteq S_a$, il che banalmente implica che se $S_a = \emptyset$, allora anche $S_{ott} = \emptyset$. Inoltre, si dimostra la seguente osservazione.

Osservazione 8 *Se S_{ott} è un insieme finito e non vuoto, S_{ott} contiene un solo punto.*

Dimostrazione Ragioniamo per assurdo. Supponiamo che S_{ott} sia un insieme finito e contenga più di un punto. Siano $x_1, x_2 \in S_{ott}$, $x_1 \neq x_2$, due punti distinti di S_{ott}. Si dovrà avere $cx_1 = cx_2$. Per la convessità di S_a si ha che tutti i punti:

$$\lambda x_1 + (1-\lambda)x_2 \quad \lambda \in (0,1),$$

(i punti lungo il segmento che congiunge x_1 e x_2), appartengono a S_a. Inoltre, la linerità della funzione obiettivo implica:

$$c[\lambda x_1 + (1-\lambda)x_2] = \lambda cx_1 + (1-\lambda)cx_2 = \lambda cx_1 + (1-\lambda)cx_1 = cx_1 \quad \forall\ \lambda \in (0,1).$$

Ma allora tutto il segmento che congiunge x_1 e x_2 è contenuto in S_{ott}. Essendo tale insieme costituito da un numero infinito di punti, questo contraddice

l'ipotesi di finitezza dell'insieme S_{ott}.

Prima di addentrarci nell'analisi di S_{ott} introduciamo un metodo di tipo grafico che ci consentirà di determinare S_{ott} nel caso di problemi con due sole variabili. Nonostante la limitata applicabilità di questo metodo di risoluzione (i problemi reali hanno tipicamente molto più di due sole variabili), esso ci consentirà di visualizzare facilmente le diverse possibili forme dell'insieme S_{ott}.

3.3.1 Metodo di risoluzione grafica

Per descrivere la risoluzione grafica consideriamo il seguente esempio:

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Per prima cosa disegniamo la regione ammissibile S_a. Poi prendiamo la funzione obiettivo e poniamola uguale a 0, cioè consideriamo $cx = 0$ nel caso generale e $x_1 + 2x_2 = 0$ nell'esempio. Questa è una retta che passa per l'origine che contiene tutti i punti in R^2 con valore della funzione obiettivo pari a 0. Ora voglio individuare qual è la direzione di crescita del fascio di rette

$$cx = k, \quad k \in R$$

parallele alla retta $cx = 0$ passante per l'origine. Nell'esempio avremo

$$x_1 + 2x_2 = k, \quad k \in R$$

fascio di rette parallele a $x_1 + 2x_2 = 0$. Possiamo individuare la direzione di crescita per esempio tracciando la retta $cx = 1$ (quindi $x_1 + 2x_2 = 1$ nel nostro esempio) e la direzione di crescita sarà quella che va dalla retta $cx = 0$ verso la retta $cx = 1$. In Figura 3.1 la direzione di crescita per il nostro esempio è indicata con una freccia sulla retta $x_1 + 2x_2 = 0$. A questo punto sono possibili due casi:

Caso 1 Muovendomi dalla retta $cx = 0$ verso la direzione di crescita ho almeno una retta del fascio con intersezione non vuota con S_a. In tal caso abbiamo due sottocasi possibili.

- **Caso 1.1** Esiste un valore $\overline{k}$ tale che la retta $cx = \overline{k}$ ha intersezione non vuota con S_a mentre tutte le retta $cx = k$ per $k > \overline{k}$ hanno intersezione vuota con S_a. In tal caso $\overline{k}$ è il valore ottimo del problema e l'intersezione della retta $cx = \overline{k}$ con S_a costituisce l'insieme S_{ott}.
- **Caso 1.2** Esiste un $K \geq 0$ tale che per ogni $k \geq K$ la retta $cx = k$ ha intersezione non vuota con S_a. In tal caso ci troviamo nella situazione in cui $S_{ott} = \emptyset$ in quanto il problema ha obiettivo illimitato.

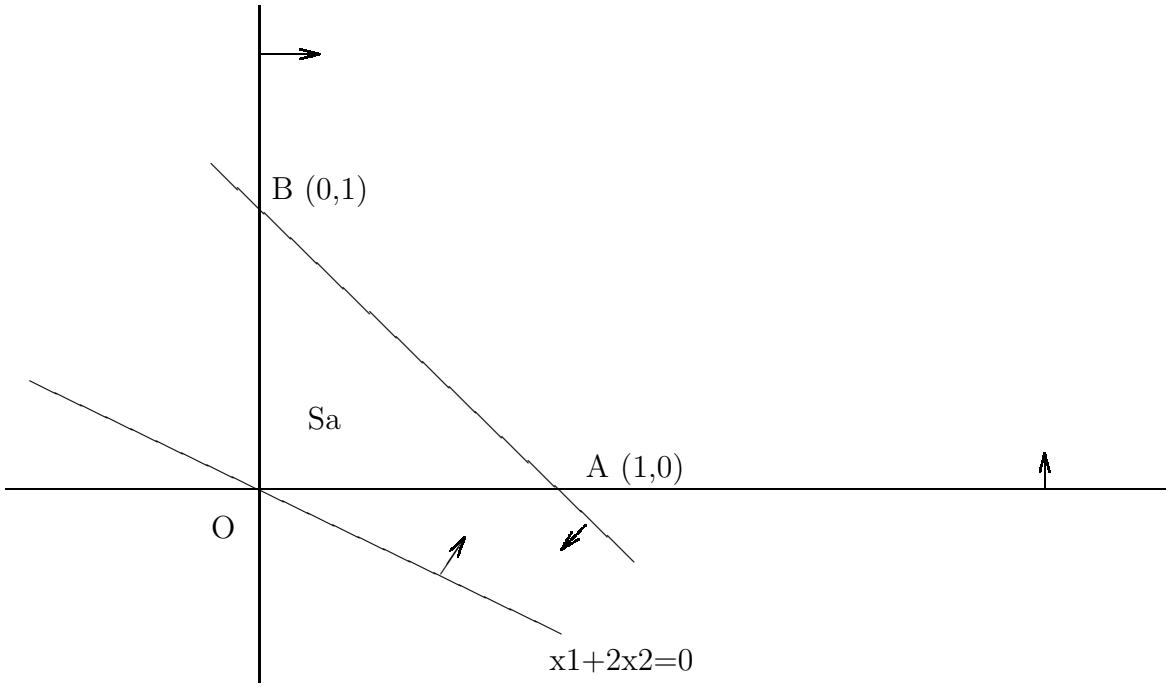


Figura 3.1:

Caso 2 Muovendomi dalla retta $cx = 0$ verso la direzione di crescita non ho alcuna retta del fascio con intersezione non vuota con S_a. In tal caso mi muovo nella direzione di decrescita e mi arresto con la prima retta $cx = \overline{k}$ ($\overline{k} < 0$) che ha intersezione non vuota con S_a (quindi per ogni $k > \overline{k}$ si ha che la retta $cx = k$ ha intersezione vuota con S_a). Il valore $\overline{k}$ è il valore ottimo del problema e l'intersezione della retta $cx = \overline{k}$ con S_a rappresenta l'insieme S_{ott}.

Nel nostro esempio si può vedere che ci si trova nel Sottocaso 1.1 con $\overline{k} = 2$ e S_{ott} ristretto al solo punto B di coordinate $(0, 1)$.

3.3.2 Le diverse forme possibili di S_{ott}

Siamo ora pronti a individuare tutte le forme possibili di S_{ott} al variare di S_a.

Caso 1 $S_a = \emptyset$. In tal caso, essendo S_{ott} un sottinsieme di S_a, può solo essere $S_{ott} = \emptyset$.

Caso 2 $S_a \neq \emptyset$ e politopo. Un politopo è un insieme chiuso e limitato, mentre la funzione obiettivo è lineare e quindi certamente continua. Di conseguenza, il Teorema di Weierstrass garantisce l'esistenza di almeno una soluzione ottima, ovvero $S_{ott} \neq \emptyset$. Sono possibili due sottocasi.

Caso 2.1 S_{ott} è costituito da un solo punto.

Caso 2.2 S_{ott} è costituito da un insieme infinito e limitato di punti.

Si noti che l'Osservazione 8 esclude la possibilità di un numero finito e maggiore di 1 di punti in S_{ott}, mentre la limitatezza di S_{ott} è garantita dal fatto che è un sottinsieme di S_a che a sua volta è un politopo e quindi è limitato.

Caso 3 $S_a \neq \emptyset$ e poliedro illimitato. Sono possibili quattro sottocasi.

Caso 3.1 $S_{ott} = \emptyset$ in quanto l'obiettivo è illimitato, ovvero esiste una sequenza infinita di punti $\{x_k\}$ di S_a lungo cui la funzione obiettivo cresce a $+\infty$. Formalmente:

$$\exists\ \{x_k\}\ :\ x_k \in S_a\ \forall\ k\ \text{ e }\ cx_k \to +\infty\ \ k \to +\infty.$$

Caso 3.2 S_{ott} è costituito da un solo punto.

Caso 3.3 S_{ott} è costituito da un insieme infinito e limitato di punti.

Caso 3.4 S_{ott} è costituito da un insieme infinito e illimitato di punti.

Come esercizio si applichi ora la risoluzione grafica ai seguenti esempi riconoscendo in essi molti dei casi possibili per S_{ott} precedentemente elencati.

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & -x_1 + x_2 \leq 0 \\ & x_1 - x_2 \leq 1 \\ & x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & -x_1 \\ & -x_1 + x_2 \leq 0 \\ & x_1 - x_2 \leq 1 \\ & x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & -x_2 \\ & -x_1 + x_2 \leq 0 \\ & x_1 - x_2 \leq 1 \\ & x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & x_1 - x_2 \\ & -x_1 + x_2 \leq 0 \\ & x_1 - x_2 \leq 1 \\ & x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

3.4 Il Teorema Fondamentale della PL

La risoluzione grafica dei precedenti esempi mostra anche che, quando $S_{ott} \neq \emptyset$, tale insieme contiene sempre almeno un vertice. Non è un caso. Vale infatti un teorema molto importante noto come Teorema Fondamentale della Programmazione Lineare. Prima di dimostrare questo abbiamo bisogno di dimostrare un lemma.

Lemma 1 *Dato un problema di PL in forma canonica, se $S_{ott} \neq \emptyset$, allora per ogni raggio estremo r di S_a si ha:*

$$cr \leq 0.$$

Dimostrazione La dimostrazione è per assurdo. Supponiamo infatti che esista un raggio estremo r tale che

$$cr > 0 \tag{3.1}$$

Notiamo che $S_{ott} \neq \emptyset$ implica $S_a \neq \emptyset$. Sia allora $x_0 \in S_a$. Poichè r è un raggio, in base alla Definizione 7 di raggio avremo che

$$\forall\ \lambda \geq 0\ :\quad x_0 + \lambda r \in S_a.$$

Calcoliamo ora il valore della funzione obiettivo nei punti $x_0 + \lambda r$:

$$c(x_0 + \lambda r) = cx_0 + \lambda cr$$

Ma, in base a (3.1) possiamo concludere che

$$c(x_0 + \lambda r) \to +\infty \qquad \lambda \to +\infty,$$

cioè $S_{ott} = \emptyset$ in quanto l'obiettivo è illimitato sulla regione ammissibile, il che contraddice $S_{ott} \neq \emptyset$.

Si noti che la dimostrazione del lemma ci dice anche che qualora esista un raggio estremo r di S_a tale che $cr > 0$, allora il problema ha obiettivo illimitato. Si può dimostrare anche il viceversa, cioè se il problema ha obiettivo illimitato, allora esiste sicuramente un raggio estremo r di S_a tale che $cr > 0$. Siamo ora pronti a dimostrare il Teorema Fondamentale della PL.

Teorema 3 *Dato un problema di PL in forma canonica, se $S_{ott} \neq \emptyset$, allora S_{ott} contiene almeno un vertice di S_a.*

Dimostrazione Indichiamo con $v_1, \ldots, v_k$ i vertici di S_a e, nel caso in cui S_a sia un poliedro illimitato, indichiamo con $r_1, \ldots, r_h$ i raggi estremi di S_a. Se $S_{ott} \neq \emptyset$, sia $x^* \in S_{ott}$.
Utilizzeremo una *dimostrazione per assurdo*. Per assurdo supponiamo che

$$v_1, \ldots, v_k \notin S_{ott}$$

cioè supponiamo che nessun vertice di S_a appartenga a S_{ott}. In particolare avremo

$$cv_i < cx^* \quad i = 1, \ldots, k \tag{3.2}$$

In base al Teorema 2, poiché $x^* \in S_a$ avremo che

$$\exists\ \lambda_1^*, \ldots, \lambda_k^* \geq 0, \quad \sum_{i=1}^{k} \lambda_i^* = 1, \quad \exists\ \mu_1^*, \ldots, \mu_h^* \geq 0 \tag{3.3}$$

tali che

$$x^* = \sum_{i=1}^{k} \lambda_i^* v_i + \sum_{j=1}^{h} \mu_j^* r_j.$$

Quindi avremo

$$cx^* = c\left[\sum_{i=1}^{k} \lambda_i^* v_i + \sum_{j=1}^{h} \mu_j^* r_j\right]$$

e per la linearità della funzione obiettivo

$$cx^* = \sum_{i=1}^{k} \lambda_i^* (cv_i) + \sum_{j=1}^{h} \mu_j^* (cr_j).$$

Nel Lemma 1 abbiamo dimostrato che:

$$cr_j \leq 0 \quad j = 1, \ldots, h. \tag{3.4}$$

Ora, in base a (3.3) e (3.4) avremo

$$cx^* = \sum_{i=1}^{k} \lambda_i^* (cv_i) + \sum_{j=1}^{h} \underbrace{\mu_j^*}_{\geq 0} \underbrace{(cr_j)}_{\leq 0}.$$

da cui

$$cx^* \leq \sum_{i=1}^{k} \lambda_i^* (cv_i).$$

Ma ora possiamo sfruttare (3.2):

$$cx^* \leq \sum_{i=1}^{k} \lambda_i^* \underbrace{(cv_i)}_{<cx^*} < \sum_{i=1}^{k} \lambda_i^* (cx^*).$$

(si noti che lo strettamente minore vale perché almeno uno dei λ_i^* è strettamente positivo in quanto, in base a (3.3), la loro somma deve essere pari a 1). Poiché cx^* non dipende dall'indice i della sommatoria, lo possiamo portare fuori dalla stessa e quindi

$$cx^* < (cx^*) \sum_{i=1}^{k} \lambda_i^*.$$

Ma ora in base a (3.3) abbiamo che $\sum_{i=1}^{k} \lambda_i^* = 1$, da cui

$$cx^* < cx^*$$

il che è assurdo.

Questo risultato è alla base della procedura di risoluzione che descriveremo, l'algoritmo del *simplesso*. Infatti, tale algoritmo ricerca la soluzione ottima cercando di spostarsi ad ogni iterazione in modo intelligente da un vertice all'altro di S_a. Per modo intelligente si intende che l'algoritmo tenta di spostarsi ad una data iterazione da un vertice a uno con valore della funzione obiettivo maggiore.

3.5 Preparazione al metodo del simplesso

Sappiamo che il nostro scopo è trovare almeno un punto in S_{ott} oppure stabilire che $S_{ott} = \emptyset$ in quanto anche $S_a = \emptyset$ oppure perché l'obiettivo del problema di PL è illimitato. Dobbiamo allora individuare un metodo che ci consenta di raggiungere il nostro scopo. Abbiamo già incontrato un tale metodo, quello di risoluzione grafica, la cui applicabilità è però ristretta ai soli problemi con due variabili. Quello che vogliamo ora è un metodo che possa essere applicato con un numero qualsiasi di variabili. Questo metodo sarà il *metodo del simplesso*. Prima però di arrivare a descriverlo avremo bisogno di alcuni passaggi intermedi. Per prima cosa ci spostiamo dalla forma canonica a un'altra forma particolare dei problemi di PL.

3.5.1 I problemi di PL in forma standard

I problemi di PL in forma *standard* hanno la seguente formulazione:

$$\begin{aligned} \max \quad & cx \\ & a_i x = b_i \quad i = 1, \ldots, m \\ & x \geq 0 \end{aligned}$$

o, equivalentemente, in forma matriciale:

$$\begin{aligned} \max \quad & cx \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

dove A è la matrice la cui i-esima riga è il vettore a_i e b è il vettore la cui i-esima componente è b_i. Rispetto alla forma canonica cambia solamente il fatto che i vincoli non sono più di $\leq$ ma sono di uguaglianza. Vale la seguente osservazione.

Osservazione 9 *Ogni problema di PL in forma canonica può essere trasformato in uno equivalente in forma standard.*

Dimostrazione Sia dato il problema di PL in forma canonica

$$\begin{aligned} \max \quad & cx \\ & a_i x \leq b_i \quad i = 1, \ldots, m \\ & x \geq 0 \end{aligned}$$

Con l'aggiunta di una nuova variabile y_i per ogni vincolo $a_i x \leq b_i$, possiamo esprimere tale vincolo attraverso la seguente coppia di vincoli:

$$a_i x + y_i = b_i, \quad y_i \geq 0.$$

Quindi, il problema di PL in forma canonica è equivalente al seguente:

$$\begin{aligned} \max \quad & cx \\ & a_i x + y_i = b_i \quad i = 1, \ldots, m \\ & x \geq 0 \\ & y_i \geq 0 \qquad i = 1, \ldots, m \end{aligned}$$

che è in forma standard (vincoli di uguaglianza e variabili non negative).

Avendo già dimostrato in precedenza che ogni problema di PL può essere ricondotto ad uno equivalente in forma canonica, l'osservazione sopra ci dice anche che ogni problema di PL può essere ricondotto ad uno equivalente in forma standard.

Riguardo la matrice $A \in R^{m \times n}$ con i-esima riga a_i, nel seguito faremo sempre la seguente ipotesi: *la matrice A ha rango pari a m, il numero delle sue righe*, dove ricordiamo che il rango di una matrice coincide con il massimo numero di righe (o, equivalentemente, di colonne) linearmente indipendenti della matrice. Si noti che deve necessariamente essere $m \leq n$ (per $n < m$ il rango di A potrebbe essere al più n e non potrebbe essere pari a m). Si può dimostrare che anche questa sul rango non è una condizione restrittiva e che ci si può sempre ricondurre a essa, eliminando eventuali vincoli ridondanti (si veda la Sezione 4.2).

3.5.2 Basi e soluzioni di base

Un concetto importante è quello di base di un problema di PL in forma standard.

Definizione 8 *Si definisce* base *di un problema di PL in forma standard un sottinsieme:*

$$B = \{x_{i_1}, \ldots, x_{i_m}\}$$

di m delle n variabili del problema di PL con la proprietà che la matrice $A_B \in R^{m \times m}$ *ottenuta considerando le sole colonne di A relative alle variabili* x_{i_k}, $k = 1, \ldots, m$, *sia invertibile. Le variabili dell'insieme B verranno dette* variabili in base*, quelle al di fuori di B verranno raggruppate nell'insieme:*

$$N = \{x_{i_{m+1}}, \ldots, x_{i_n}\}$$

e verranno dette variabili fuori base*.*

Esempio 14 *Sia dato il seguente problema di PL in forma standard:*

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 + 2x_3 + 2x_4 + x_5 \\ & x_1 + 2x_2 + 2x_3 + x_4 - x_5 = 2 \\ & x_1 + 2x_2 + x_3 + 4x_4 - 2x_5 = 2 \\ & x1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

In questo caso si hanno due vincoli e quindi $m = 2$*. Se prendiamo* $B_1 = \{x_1, x_2\}$ *vediamo che* B_1 *non è una base. Si ha infatti:*

$$A_{B_1} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

che non è invertibile. Invece, $B_2 = \{x_1, x_3\}$ *è una base in quanto:*

$$A_{B_2} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

è invertibile. Allo stesso modo si verifichi che $B_3 = \{x_3, x_4\}$ *e* $B_4 = \{x_4, x_5\}$ *sono basi.*

Introduciamo ora il concetto di *soluzione di base*. Data una base B, indichiamo con $x_B \in R^m$ il vettore delle variabili in base, con $x_N \in R^{n-m}$ il vettore delle variabili fuori base, $c_B \in R^m$ il vettore dei costi relativi alle variabili in base, con $c_N \in R^{n-m}$ il vettore dei costi relativi alle variabili fuori base e, infine, con $A_N \in R^{m \times (n-m)}$ la matrice ottenuta da A considerando le sole colonne relative alle variabili fuori base. Possiamo ora riscrivere il problema di PL in forma standard nella seguente forma equivalente:

$$\begin{aligned} \max \quad & c_B x_B + c_N x_N \\ & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{aligned}$$

e quindi anche in questo modo:

$$\begin{aligned} \max \quad & c_B x_B + c_N x_N \\ & A_B x_B = b - A_N x_N \\ & x_B, x_N \geq 0 \end{aligned}$$

Moltiplichiamo ora i vincoli per A_B^{-1}. Si ottiene:

$$\begin{aligned} \max \quad & c_B x_B + c_N x_N \\ & x_B = A_B^{-1} b - A_B^{-1} A_N x_N \\ & x_B, x_N \geq 0 \end{aligned}$$

In pratica questo coincide con la risoluzione di un sistema con m incognite e m vincoli, dove le incognite sono le variabili nella base B, mentre le variabili fuori base (quelle in N) vengono trattate come parametri. Infine, sostituendo x_B nell'obiettivo si ottiene la seguente ulteriore riformulazione, sempre equivalente alle precedenti:

$$\begin{aligned} \max \quad & c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N \\ & x_B = A_B^{-1} b - A_B^{-1} A_N x_N \\ & x_B, x_N \geq 0 \end{aligned} \tag{3.5}$$

Questa riformulazione viene detta *riformulazione del problema di PL rispetto alla base B*. Siamo ora pronti a dare la definizione di soluzione di base.

Definizione 9 *Si definisce* soluzione di base *associata alla base B, la seguente soluzione ottenuta ponendo $x_N = 0$ nei vincoli in (3.5):*

$$x_B = A_B^{-1} b \qquad x_N = 0.$$

Se $A_B^{-1} b \geq 0$ la soluzione di base si dice ammissibile*. Se inoltre si ha $A_B^{-1} b > 0$ si parla di soluzione di base* non degenere*, altrimenti si parla di soluzione di base* degenere*.*

Ma vediamo di tornare all'Esempio 14.

Esempio 15 *Data la base B_2 dell'Esempio 14, si ha*

$$A_{B_2}^{-1} = \begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix} \qquad A_{N_2} = \begin{bmatrix} 2 & 1 & -1 \\ 2 & 4 & -2 \end{bmatrix}$$

da cui si ottiene la seguente riformulazione rispetto alla base B_2:

$$\begin{aligned} \max \quad & 6 - 2x_2 - 13x_4 + 8x_5 \\ & x_1 = 2 - 2x_2 - 7x_4 + 3x_5 \\ & x_3 = 0 + 3x_4 - x_5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Ponendo a 0 le variabili fuori base si ottiene la seguente soluzione di base associata a B_2:

$$x_1 = 2 \quad x_3 = 0 \quad x_2 = x_4 = x_5 = 0.$$

La soluzione di base è ammissibile (tutte le variabili hanno valore non negativo) e degenere (la variabile in base x_3 ha valore nullo). Il valore dell'obiettivo in

corrispondenza di tale soluzione di base è 6 (lo si vede ponendo a 0 le variabili fuori base nell'obiettivo).
In modo analogo si dimostri che la soluzione di base associata a $B_3 = \{x_3, x_4\}$ *è:*

$$x_3 = 6/7 \quad x_4 = 2/7 \quad x_1 = x_2 = x_5 = 0,$$

che è ammissiibile e non degenere e che la soluzione di base associata a $B_4 = \{x_4, x_5\}$ *è:*

$$x_4 = -1 \quad x_5 = -3 \quad x_1 = x_2 = x_3 = 0,$$

che è non ammissibile.

Vale la seguente osservazione.

Osservazione 10 *Data una soluzione di base ammissibile e non degenere esiste un'unica base che la rappresenta, mentre una soluzione di base ammissibile e degenere è rappresentata da più basi.*

Si verifichi, ad esempio, che la base $B_5 = \{x_1, x_4\}$ ha come soluzione di base associata:

$$x_1 = 2 \quad x_4 = 0 \quad x_2 = x_3 = x_5 = 0,$$

che è la stessa associata alla base B_2.

In precedenza abbiamo stabilito che, se $S_{ott} \neq \emptyset$, allora almeno un punto di S_{ott} è un vertice di S_a, il che ci consente di restringere la ricerca delle soluzioni ottime ai soli vertici di S_a. Ma cosa ha a che fare tutto questo con le basi e le soluzioni di base associate? La risposta ci viene dalla seguente osservazione.

Osservazione 11 *L'insieme dei vertici di* S_a *coincide con l'insieme delle soluzioni di base ammissibili del problema di PL.*

Quindi questo ci dice che possiamo indifferentemente parlare di vertici e di soluzioni di base ammissibili e possiamo ricercare soluzioni ottime del problema restringendo l'attenzione alle sole soluzioni di base ammissibili.
Diamo ora la definizione di basi adiacenti.

Definizione 10 *Due basi* B' *e* B'' *si definiscono* adiacenti *se hanno* $m - 1$ *variabili uguali e differiscono per una sola variabile.*

Nell'esempio le basi B_3 e B_4, che differiscono per una sola variabile, sono adiacenti. Il concetto di adiacenza si estende anche alle soluzioni di base.

Definizione 11 *Due soluzioni di base distinte si definiscono adiacenti se esistono due basi* B' *e* B'' *che le rappresentano e che sono tra loro adiacenti.*

Si noti che due basi adiacenti non corrispondono necessariamente a due soluzioni di base adiacenti. Esse infatti possono corrispondere alla stessa soluzione di base come accade, ad esempio, con le basi B_2 e B_5.

3.5.3 L'operazione di cardine

Vogliamo ora introdurre una procedura, detta *operazione di cardine*, che ci consenta di passare dalla riformulazione del problema di PL rispetto ad una base B alla riformulazione rispetto ad una base adiacente B'. Sia data la base:

$$B = \{x_{i_1}, \ldots, x_{i_m}\}.$$

Sia

$$N = \{x_{i_{m+1}}, \ldots, x_{i_n}\}$$

l'insieme delle variabili fuori base. Abbiamo visto che, data la base B, la riformulazione del problema di PL rispetto alla base B è data da (3.5). Indicando con:

- γ_0 il valore $c_B A_B^{-1} b$;
- γ_j, $j = 1, \ldots, n-m$, le componenti del vettore
$$c_N - c_B A_B^{-1} A_N;$$
- β_r, $r = 1, \ldots, m$, le componenti del vettore $A_B^{-1} b$;
- α_{rj}, $r = 1, \ldots, m$, $j = 1, \ldots, n-m$, le componenti della matrice $-A_B^{-1} A_N$

possiamo riscrivere la riformulazione (3.5) rispetto alla base B nel seguente modo:

$$\begin{array}{rc} \max & \gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}} \\ & x_{i_1} = \beta_1 + \sum_{j=1}^{n-m} \alpha_{1j} x_{i_{m+j}} \\ & \cdots \\ & x_{i_k} = \beta_k + \sum_{j=1}^{n-m} \alpha_{kj} x_{i_{m+j}} \\ & \cdots \\ & x_{i_m} = \beta_m + \sum_{j=1}^{n-m} \alpha_{mj} x_{i_{m+j}} \\ & x_1, \ldots, x_n \geq 0 \end{array} \tag{3.6}$$

Supponiamo ora di voler passare dalla base B alla base adiacente B' ottenuta rimuovendo da B la variabile x_{i_k}, $1 \leq k \leq m$, e sostituendola con la variabile fuori base $x_{i_{m+h}}$, $1 \leq h \leq n-m$, ovvero:

$$B' = \{x_{i_1}, \ldots, x_{i_{k-1}}, x_{i_{m+h}}, x_{i_{k+1}}, \ldots, x_{i_m}\}.$$

La prima domanda che ci dobbiamo porre è quando B' è effettivamente una base. Perché lo sia si deve avere che $A_{B'}$ è invertibile. Tuttavia il seguente risultato ci consente una verifica molto più rapida.

Osservazione 12 *Si ha che $A_{B'}$ è invertibile e quindi B' è una base se e solo se nella riformulazione associata alla base B il coefficiente di $x_{i_{m+h}}$ nell'equazione relativa a x_{i_k} è diverso da 0, ovvero se e solo se:*

$$\alpha_{kh} \neq 0.$$

Supposto che $\alpha_{kh} \neq 0$, vediamo ora di passare dalla riformulazione rispetto alla base B a quella rispetto alla base B'. Per fare questo dovremo compiere le seguenti operazioni.

- Ricavare $x_{i_{m+h}}$ dall'equazione relativa a x_{i_k}, cioè:

$$x_{i_{m+h}} = -\frac{\beta_k}{\alpha_{kh}} + \frac{1}{\alpha_{kh}} x_{i_k} - \sum_{j=1,\ j\neq h}^{n-m} \frac{\alpha_{kj}}{\alpha_{kh}} x_{i_{m+j}}. \tag{3.7}$$

- sostituire ogni occorrenza della variabile $x_{i_{m+h}}$ nelle restanti equazioni e nell'obiettivo con la parte destra di (3.7).

Una volta eseguite queste operazioni si ha la seguente riformulazione rispetto a B':

$$\begin{array}{rc} \max & \left(\gamma_0 - \gamma_h \frac{\beta_k}{\alpha_{kh}}\right) + \frac{\gamma_h}{\alpha_{kh}} x_{i_k} + \sum_{j=1,\ j\neq h}^{n-m} \left(\gamma_j - \gamma_h \frac{\alpha_{kj}}{\alpha_{kh}}\right) x_{i_{m+j}} \\ x_{i_1} = \left(\beta_1 - \alpha_{1h} \frac{\beta_k}{\alpha_{kh}}\right) + \frac{\alpha_{1h}}{\alpha_{kh}} x_{i_k} + \sum_{j=1,\ j\neq h}^{n-m} \left(\alpha_{1j} - \alpha_{1h} \frac{\alpha_{kj}}{\alpha_{kh}}\right) \\ \cdots \\ x_{i_{m+h}} = -\frac{\beta_k}{\alpha_{kh}} + \frac{1}{\alpha_{kh}} x_{i_k} - \sum_{j=1,\ j\neq h}^{n-m} \frac{\alpha_{kj}}{\alpha_{kh}} x_{i_{m+j}}. \\ \cdots \\ x_{i_m} = \left(\beta_m - \alpha_{mh} \frac{\beta_k}{\alpha_{kh}}\right) + \frac{\alpha_{mh}}{\alpha_{kh}} x_{i_k} + \sum_{j=1,\ j\neq h}^{n-m} \left(\alpha_{mj} - \alpha_{mh} \frac{\alpha_{kj}}{\alpha_{kh}}\right) \\ x_1, \ldots, x_n \geq 0 \end{array} \tag{3.8}$$

NOTA BENE Per poter recuperare dalle riformulazioni alcune informazioni (nel seguito vedremo, ad esempio, come sfruttare la riformulazione rispetto a una base B per poter ottenere la matrice A_B^{-1}) è necessario mantenere un ordine tra le variabili in una base. Quindi se nella base B' la variabile $x_{i_{m+h}}$ sostituisce la variabile x_{i_k} a cui corrisponde la k-esima equazione della riformulazione rispetto a B, nella riformulazione rispetto a B' l'equazione relativa alla variabile $x_{i_{m+h}}$ dovrà ancora essere la k-esima, mentre la posizione delle equazioni relative a tutte le altre variabili deve rimanere invariata rispetto alla precedente riformulazione.

Vediamo ora di chiarire meglio come si esegue l'operazione di cardine operando sul nostro esempio. Consideriamo la base B_2. Come già visto, la riformulazione rispetto a questa base è la seguente:

$$\begin{array}{rc} \max & 6 - 2x_2 - 13x_4 + 8x_5 \\ & x_1 = 2 - 2x_2 - 7x_4 + 3x_5 \\ & x_3 = 0 + 3x_4 - x_5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array}$$

Supponiamo ora di voler passare alla riformulazione rispetto alla base adiacente $B_6 = \{x_1, x_5\}$ ottenuta rimuovendo da B_2 la variabile x_3 e sostituendola con

x_5. Per prima cosa notiamo che il coefficiente di x_5 nell'equazione:

$$x_3 = 0 + 3x_4 - x_5$$

relativa a x_3 è pari a $-1 \neq 0$ e quindi B_6 è una base in base all'Osservazione 12. Per prima cosa dobbiamo ricavare un'equazione per x_5 partendo dall'equazione per x_3 riportata appena sopra. Si ottiene:

$$x_5 = 0 - x_3 + 3x_4.$$

Quindi, dobbiamo sostituire a x_5 nelle equazioni restanti e nell'obiettivo la parte destra di tale equazione. Avremo quindi nell'unica equazione restante:

$$x_1 = 2 - 2x_2 - 7x_4 + 3(0 - x_3 + 3x_4),$$

e nell'obiettivo:

$$6 - 2x_2 - 13x_4 + 8(0 - x_3 + 3x_4).$$

Svolgendo i calcoli la riformulazione rispetto a B_6 sarà la seguente:

$$\begin{aligned} \max \quad & 6 - 2x_2 - 8x_3 + 11x_4 \\ & x_1 = 2 - 2x_2 - -3x_3 + 2x_4 \\ & x_5 = 0 - x_3 + 3x_4 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Si noti che la posizione dell'equazione relativa a x_5 nelle nuova riformulazione è la stessa dell'equazione relativa a x_3 nella precedente riformulazione, mentre la posizione delle equazioni relative alle altri variabili (in questo caso la sola x_1) rimane invariata.
Una volta chiarita l'operazione di cardine siamo pronti ad addentrarci nel prossimo capitolo nella descrizione del metodo del simplesso per risolvere problemi di PL.

Capitolo 4

L'algoritmo del simplesso

In questo capitolo presenteremo un algoritmo per la risoluzione dei problemi di PL, l'algoritmo del simplesso. Più precisamente vedremo una variante di tale algoritmo (un'altra la vedremo nel Capitolo 5).

4.1 I passi dell'algoritmo

Per la coincidenza tra vertici e soluzioni di base ammissibili, abbiamo visto che possiamo restringere la ricerca delle soluzioni ottime alle sole soluzioni di base ammissibili. Per questa ragione l'algoritmo del simplesso procede passando a ogni iterazione da una base ammissibile a una adiacente ancora ammissibile fino a quando è sodisfatta una qualche condizione di terminazione. Il passaggio da una base all'altra viene fatto in modo intelligente: dal momento che vogliamo massimizzare il nostro obiettivo, desideriamo passare da una base ammissibile con una certa soluzione di base associata ad un'altra base ammissibile con una soluzione di base che ha valore dell'obiettivo migliore (più elevato) o quanto meno non peggiore rispetto a quella precedente.
Vedremo nel seguito come questo sia verificato. Supponiamo ora di avere una base ammissibile B con la relativa riformulazione (3.6). Si noti che problemi non banali sono stabilire se *esiste una base ammissibile* (o, equivalentemente, stabilire se $S_a \neq \emptyset$) e, nel caso esista, determinarne una. Di questi problemi ci occuperemo in seguito. Per il momento supponiamo di avere già a disposizione una base ammissibile B (ovvero $\beta_k \geq 0, k = 1, \ldots, m$, in (3.6)).

4.1.1 Verifica di ottimalità

La prima domanda che ci poniamo è la seguente: quando possiamo dire che la soluzione di base ammissibile associata a B è una soluzione ottima del nostro problema? A questo proposito una particolare rilevanza hanno i coefficienti delle variabili fuori base nell'obiettivo della riformulazione (3.6). Questi vengono detti anche *coefficienti di costo ridotto delle variabili fuori base* e sono

interpretabili come indicazione della variazione dell'obiettivo in corrispondenza dell'incremento di un'unità della variabile fuori base corrispondente. Infatti, se consideriamo l'obiettivo della riformulazione (3.6):

$$\gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}},$$

supponiamo di tenere a 0 il valore di tutte le variabili fuori base tranne la variabile $x_{i_{m+h}}$ il cui valore viene incrementato a 1. Il nuovo valore dell'obiettivo è $\gamma_0 + \gamma_h$ con una variazione rispetto al valore γ_0 pari proprio al valore γ_h del coefficiente di costo ridotto γ_h. Ma in che modo i coefficienti di costo ridotto ci possono dire se la soluzione di base associata alla base B è una soluzione ottima? Una condizione *sufficiente* per garantire l'ottimalità è la seguente:

$$\gamma_j \leq 0 \quad j = 1, \ldots, n-m. \tag{4.1}$$

Si richiede quindi che i coefficienti di costo ridotto delle variabili fuori base siano tutti non positivi. Ricordando che γ_j sono le componenti del vettore $\mathbf{c}_N - \mathbf{c}_B \mathbf{A}_B^{-1} \mathbf{A}_N$ (detto anche, per questa ragione, vettore dei coefficienti di costo ridotto), in forma vettoriale la condizione sufficiente di ottimalità (4.1) si esprime nel modo seguente:

$$\mathbf{c}_N - \mathbf{c}_B \mathbf{A}_B^{-1} \mathbf{A}_N \leq 0. \tag{4.2}$$

Ma vediamo di capire perché la condizione (4.1) ci garantisce che la soluzione di base associata a B è una soluzione ottima del problema. Sappiamo che il valore dell'obiettivo in corrispondenza di questa soluzione di base è γ_0. Notiamo inoltre che in S_a si ha $x_{i_{m+j}} \geq 0$, $j = 1, \ldots, n-m$. Quindi per il valore dell'obiettivo in S_a si avrà:

$$\gamma_0 + \sum_{j=1}^{n-m} \underbrace{\gamma_j}_{\leq 0} \underbrace{x_{i_{m+j}}}_{\geq 0} \leq \gamma_0, \tag{4.3}$$

ovvero in S_a il valore dell'obiettivo non può mai superare il valore γ_0. Essendo questo anche il valore dell'obiettivo per la nostra soluzione di base, tale soluzione di base è anche soluzione ottima del nostro problema.
È importante sottolineare che la condizione è solo sufficiente, cioè può succedere che la soluzione di base sia già una soluzione ottima ma la condizione (4.1) non sia soddisfatta. Ciò però può accadere *solo* nel caso di una soluzione di base degenere.

Esempio 16 *Sia data la seguente riformulazione rispetto alla base* $\{x_3, x_4\}$ *di un problema di PL:*

$$\begin{aligned} \max \quad & x_1 \\ & x_3 = 1 - x_2 \\ & x_4 = -x_1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

La soluzione di base corrispondente è:

$$x_3 = 1 \quad x_4 = 0 \quad x_1 = x_2 = 0.$$

Si noti che è degenere. La condizione (4.1) non è soddisfatta (il coefficiente di x_1 nell'obiettivo è pari a 1). Ma passiamo ora, con l'operazione di cardine, alla base adiacente $\{x_1, x_3\}$. La riformulazione rispetto a questa base è la seguente:

$$\begin{aligned} \max \quad & -x_4 \\ & x_3 = 1 - x_2 \\ & x_1 = -x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Ora la condizione sufficiente è soddisfatta e quindi la soluzione di base associata è ottima. Ma se osserviamo la soluzione di base associata, questa coincide esattamente con la precedente.

Si può comunque dimostrare che data una soluzione di base ottima *esiste sempre almeno una base corrispondente per la quale la condizione (4.1) è soddisfatta.* Nell'esempio abbiamo visto come la stessa soluzione di base sia rappresentata sia dalla base $\{x_3, x_4\}$ che dalla base $\{x_1, x_3\}$. La prima base non soddisfa la condizione (4.1), ma questa è soddisfatta dalla seconda base. Torneremo in seguito su un'altra questione e cioè quando possiamo dire che il problema ammette un'unica soluzione ottima o più soluzioni ottime.

4.1.2 Verifica di illimitatezza

Supponiamo ora che la condizione di ottimalità (4.1) non sia soddisfatta. Un'altra domanda che possiamo porci è la seguente: quando il problema ha valore dell'obiettivo illimitato? Una condizione sufficiente perché questo si verifichi è la seguente:

$$\exists\ \gamma_h > 0\ :\quad \alpha_{rh} \geq 0 \quad r = 1, \ldots, m. \tag{4.4}$$

Vediamo perché questa condizione ci garantisce che il problema ha obiettivo illimitato. Prendiamo la riformulazione (3.6) e in essa poniamo a zero tutte le variabili fuori base tranne la variabile $x_{i_{m+h}}$. Ciò che rimane è:

$$\begin{aligned} \max \quad & \gamma_0 + \gamma_h x_{i_{m+h}} \\ & x_{i_1} = \beta_1 + \alpha_{1h} x_{i_{m+h}} \\ & \ldots \\ & x_{i_m} = \beta_m + \alpha_{mh} x_{i_{m+h}} \\ & x_1, \ldots, x_n \geq 0 \end{aligned}$$

Cosa succede se faccio crescere il valore della variabile $x_{i_{m+h}}$? Si ha che per ogni $r \in \{1, \ldots, m\}$:

$$x_{i_r} = \underbrace{\beta_r}_{\geq 0} + \underbrace{\alpha_{rh}}_{\geq 0} \underbrace{x_{i_{m+h}}}_{\geq 0} \geq 0,$$

quindi per ogni possibile valore non negativo di $x_{i_{m+h}}$ le variabili in base continuano ad avere valore non negativo e quindi rimaniamo all'interno di S_a. Ma vediamo ora cosa succede all'obiettivo:

$$\gamma_0 + \underbrace{\gamma_h}_{>0} x_{i_{m+h}} \quad \underbrace{\rightarrow}_{x_{i_{m+h}} \rightarrow +\infty} \quad +\infty,$$

e quindi facendo crescere $x_{i_{m+h}}$ all'infinito non si esce mai da S_a e il valore dell'obiettivo cresce anch'esso all'infinito. Ne consegue che il problema ha $S_{ott} = \emptyset$ in quanto il valore dell'obiettivo è illimitato.

Esempio 17 *Sia data la seguente riformulazione rispetto alla base* $\{x_1, x_2\}$ *di un problema di PL:*

$$\begin{aligned} \max \quad & 2 + x_3 - x_4 \\ & x_1 = 2 + x_3 + x_4 \\ & x_2 = 1 + 2x_3 + x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Il coefficiente di x_3 *nell'obiettivo è positivo e non negativi sono anche i coefficienti di* x_3 *nelle equazioni dei vincoli. Quindi la condizione (4.4) è soddisfatta e possiamo concludere che il problema ha obiettivo illimitato. Infatti, se poniamo* $x_4 = 0$ *il problema diventa:*

$$\begin{aligned} \max \quad & 2 + x_3 \\ & x_1 = 2 + x_3 \\ & x_2 = 1 + 2x_3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

da cui si nota che facendo crescere x_3 *i valori di* x_1 *e* x_2 *continuano a mantenersi positivi (e quindi non si esce da* S_a*), mentre il valore dell'obiettivo cresce a* $+\infty$*.*

4.1.3 Scelta della variabile da far entrare in base

Se, data la base ammissibile B, non possiamo concludere che la soluzione di base associata è ottima (cioè non è soddisfatta la condizione (4.1)) e neppure che il problema ha obiettivo illimitato (cioè non è soddisfatta la condizione (4.4)), come possiamo procedere? Passeremo dalla base ammissibile B a una nuova base ammissibile B' ma facendo in modo che la soluzione di base associata a B' abbia valore dell'obiettivo migliore o quantomeno non peggiore rispetto al valore della soluzione di base associata a B. Per prima cosa individuiamo una regola per decidere come scegliere la variabile $x_{i_{m+h}}$ fuori base che dovrà entrare nella nuova base. Ricordiamo la formula dell'obiettivo:

$$\gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}}.$$

Nella soluzione di base associata a B tutte le variabili $x_{i_{m+j}}$, $j = 1, \ldots, n-m$, sono fissate a 0. Se vogliamo incrementare il valore dell'obiettivo, quali di queste variabili dovremo far crescere dal valore 0? Dovremo far crescere quelle con coefficiente di costo ridotto γ_j positivo (facendo crescere le altre il valore dell'obiettivo diminuisce oppure non cambia). Quindi dobbiamo restringere l'attenzione alle sole variabili $x_{i_{m+j}}$ tali che $\gamma_j > 0$. Qui adotteremo la regola di scelta tra queste variabili che consiste nel scegliere quella che fa crescere più rapidamente il valore dell'obiettivo e cioè la variabile $x_{i_{m+h}}$ tale che:

$$\gamma_h = \max_{j=1,\ldots,n-m} \gamma_j, \tag{4.5}$$

tenendo comunque presente che questa non è l'unica regola possibile. Nel caso il massimo sia raggiunto da più variabili adottiamo (come pura convenzione) la regola di selezionare la variabile con indice più piccolo. Vediamo ora un esempio.

Esempio 18 *Sia data la seguente riformulazione rispetto alla base* $\{x_1, x_2\}$ *di un problema di PL:*

$$\begin{aligned} \max \quad & 2 + 2x_3 + 2x_4 + x_5 \\ & x_1 = 1 - x_3 + x_4 + x_5 \\ & x_2 = 2 - x_3 - x_4 - x_5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

In questo caso tutte le variabili fuori base hanno coefficiente di costo ridotto positivo. Tra queste considero quelle il cui coefficiente di costo ridotto è massimo (la x_3 *e la* x_4*). Tra le due scelgo quella con indice minore e quindi la* x_3*. Quindi scegliamo la variabile fuori base* x_3 *come nuova variabile da far entrare in base.*

4.1.4 Scelta della variabile uscente dalla base

Una volta scelta la variabile $x_{i_{m+h}}$ che dovrà entrare in base, dobbiamo stabilire quale variabile in base dovrà farle posto, ovvero quale sarà la variabile in base x_{i_k} che dovrà uscire dalla base. Se la scelta della variabile che entra in base è guidata dal desiderio di far crescere il valore dell'obiettivo, la scelta della variabile uscente dalla base sarà motivata dal desiderio di non uscire dalla regione ammissibile. Supponiamo come in precedenza di fissare a 0 tutte le variabili fuori base tranne la $x_{i_{m+h}}$ che deve entrare in base. Si avrà dunque:

$$\begin{aligned} \max \quad & \gamma_0 + \gamma_h x_{i_{m+h}} \\ & x_{i_1} = \beta_1 + \alpha_{1h} x_{i_{m+h}} \\ & \cdots \\ & x_{i_m} = \beta_m + \alpha_{mh} x_{i_{m+h}} \\ & x_1, \ldots, x_n \geq 0 \end{aligned}$$

Fino a quando possiamo far crescere il valore di $x_{i_{m+h}}$? Abbiamo due casi:

Caso 1 Per tutti gli $r \in \{1, \ldots, m\}$ tali che $\alpha_{rh} \geq 0$ vediamo che:

$$x_{i_r} = \beta_r + \underbrace{\alpha_{rh}}_{\geq 0} \underbrace{x_{i_{m+h}}}_{\geq 0} \geq \beta_r \geq 0.$$

Quindi in questo caso non abbiamo alcuna restrizione sulla crescita di $x_{i_{m+h}}$.

Caso 2 Per gli r tali che $\alpha_{rh} < 0$, allora vediamo che il valore di $x_{i_{m+h}}$ può crescere al massimo fino a:

$$-\frac{\beta_r}{\alpha_{rh}}$$

e oltre questo valore la variabile x_{i_r} assume valori negativi (si esce quindi dalla regione ammissibile S_a).

Se vogliamo rimanere in S_a, ci dovremo arrestare non appena una variabile x_{i_r} con $\alpha_{rh} < 0$ si annulla al crescere di $x_{i_{m+h}}$. Questa sarà la variabile x_{i_k} tale che

$$\alpha_{kh} < 0 \quad \text{e} \quad -\frac{\beta_k}{\alpha_{kh}} = \min_{r \,:\, \alpha_{rh} < 0} \left\{ -\frac{\beta_r}{\alpha_{rh}} \right\}. \tag{4.6}$$

Nel caso il minimo sia raggiunto da più variabili la scelta ricade, per convenzione, su quella con indice più piccolo. La variabile x_{i_k} sarà quella che uscirà dalla base. Va ribadito come questa scelta garantisca che la nuova base B' ottenuta scambiando x_{i_k} con $x_{i_{m+h}}$ sia ancora ammissibile. Vediamo ora quale sarà la variabile uscente dalla base nell'esempio precedente.

Esempio 19 *Ricordiamo la riformulazione rispetto alla base $\{x_1, x_2\}$ del problema di PL dell'esempio:*

$$\begin{aligned} \max \quad & 2 + 2x_3 + 2x_4 + x_5 \\ & x_1 = 1 - x_3 + x_4 + x_5 \\ & x_2 = 2 - x_3 - x_4 - x_5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

In precedenza abbiamo visto che la regola di scelta della variabile fuori base che dovrà entrare in base ci porta a scegliere la x_3. Quale variabile dovrà uscire dalla base? Sia la x_1 che la x_2 sono candidate (per entrambe il coefficiente di x_3 nelle rispettive equazioni è negativo). Andiamo ora a prendere i rapporti, cambiati di segno, tra i termini noti delle equazioni e i coefficienti della x_3. Abbiamo:

$$x_1 \rightarrow \; -\frac{1}{-1} = 1 \quad x_2 \rightarrow \; -\frac{2}{-1} = 2.$$

Il minimo dei rapporti (pari a 1) è raggiunto in corrispondenza della variabile x_1 e quindi questa sarà la variabile che dovrà uscire dalla base. A conferma di

ciò notiamo che fissando a 0 tutte le variabili fuori base tranne la x_3 si ottiene:

$$\begin{aligned} \max \quad & 2 + 2x_3 \\ & x_1 = 1 - x_3 \\ & x_2 = 2 - x_3 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

e si può vedere che per mantenersi in S_a (cioè per mantenere non negative le variabili in base x_1 e x_2) possiamo far crescere x_3 al massimo fino al valore 1. In corrispondenza di tale valore si annulla la variabile x_1 e tale variabile sarà quella che dovrà uscire di base.

A questo punto, una volta selezionata la variabile entrante in base (la $x_{i_{m+h}}$) e quella uscente di base (la x_{i_k}) con le regole viste, non resta che compiere l'operazione di cardine nel modo già descritto in precedenza. Vediamo di illustrare tale operazione per il nostro esempio.

Esempio 20 *Nell'esempio x_3 deve entrare in base e deve uscire x_1. L'operazione di cardine porta alla seguente riformulazione rispetto alla nuova base $\{x_3, x_2\}$:*

$$\begin{aligned} \max \quad & 4 - 2x_1 + 4x_4 + 3x_5 \\ & x_3 = 1 - x_1 + x_4 + x_5 \\ & x_2 = 3 + x_1 - 2x_4 - 2x_5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Una volta eseguita l'operazione di cardine e passati alla nuova base B' non si dovrà fare altro che ripetere le operazioni viste sopra (verifica di ottimalità, verifica di illimitatezza, scelta della variabile entrante in base, scelta della variabile uscente dalla base, operazione di cardine) sulla nuova base B'.

4.1.5 Lo schema dell'algoritmo

A questo punto possiamo dare lo schema dell'algoritmo del simplesso mettendo assieme i pezzi studiati in precedenza.

ALGORITMO DEL SIMPLESSO

Inizializzazione Sia B_0 una base ammissibile e $k = 0$.

Passo 1- verifica ottimalità Se è soddisfatta la condizione (4.1) , la soluzione di base associata a B_k è una soluzione ottima del problema e ci si arresta. Altrimenti si vada al Passo 2.

Passo 2 - verifica di illimitatezza Se è soddisfatta la condizione (4.4), allora si ha $S_{ott} = \emptyset$ in quanto l'obiettivo del problema è illimitato e ci si arresta. Altrimenti si vada al Passo 3.

Passo 3 - scelta variabile entrante in base Si selezioni la variabile $x_{i_{m+h}}$ che dovrà entrare in base attraverso la regola (4.5).

Passo 4 - scelta variabile uscente dalla base Si selezioni la variabile x_{i_k} che dovrà uscire dalla base attraverso la regola (4.6).

Passo 5 - operazione di cardine Si generi la nuova base B_{k+1} sostituendo in B_k la variabile x_{i_k} con la variabile $x_{i_{m+h}}$ e si esegua la corrispondente operazione di cardine. Quindi, si ponga $k = k + 1$ e si ritorni al Passo 1.

Come esercizio si proceda nella risoluzione del problema dell'esempio.

4.1.6 Commenti sul metodo del simplesso

Vediamo ora di fare alcuni commenti sul metodo del simplesso.

Finitezza dell'algoritmo

Partendo da una base B, abbiamo visto che la riformulazione rispetto alla base B' (in cui $x_{i_{m+h}}$ sostituisce x_{i_k}) ottenuta attraverso l'operazione di cardine è data in (3.8). Da questa si vede immediatamente che il valore dell'obiettivo nella soluzione di base associata alla nuova base B' è

$$\gamma_0 - \gamma_h \frac{\beta_k}{\alpha_{kh}}.$$

Quando applichiamo il metodo del simplesso abbiamo che:

- $\beta_k \geq 0$ per l'ammissibilità della soluzione di base associata a B.
- $\gamma_h > 0$ per la regola di scelta (4.5) della variabile entrante in base.
- $\alpha_{kh} < 0$ per la regola di scelta (4.6) della variabile uscente dalla base.

Quindi si ha:

$$\gamma_0 - \underbrace{\gamma_h}_{>0} \frac{\overbrace{\beta_k}^{\geq 0}}{\underbrace{\alpha_{kh}}_{<0}} \geq \gamma_0.$$

Questo ci conferma che il valore dell'obiettivo nella nuova soluzione di base associata a B' è non peggiore rispetto al valore γ_0 nella soluzione di base associata a B. Inoltre, nel caso $\beta_k > 0$, il che *si verifica sempre nel caso di soluzioni di base non degeneri*, il nuovo valore dell'obiettivo è strettamente migliore rispetto al precedente. Nel caso degenere può invece succedere che i due valori siano uguali. In questo caso si può dimostrare che le due basi B e B' rappresentano la stessa soluzione di base. Quanto visto ci dice qualcosa riguardo la finitezza del metodo del simplesso, come stabilito nella seguente osservazione.

Osservazione 13 *Se tutte le soluzioni di base ammissibili in un problema di PL sono non degeneri, allora il metodo del simplesso termina in un numero finito di iterazioni.*

Dimostrazione Come abbiamo visto nel caso non degenere ad ogni iterazione la nuova soluzione di base ammissibile ha un valore strettamente migliore rispetto alla precedente e quindi è diversa da tutte quelle che la hanno preceduta, cioè tutte le soluzioni di base ammissibili associate alle basi $B_0, B_1, B_2, \ldots$ sono distinte tra loro. Essendo il numero di soluzioni di base ammissibili finito (si ricordi che queste coincidono con i vertici che sono in numero finito), il metodo dovrà arrestarsi dopo un numero finito di iterazioni o restituendo una soluzione ottima oppure stabilendo che il problema ha obiettivo illimitato.

Ma cosa succede se ci sono delle soluzioni di base ammissibili degeneri? Nel caso ci siano vertici degeneri si può verificare la situazione di *ciclaggio*. Trovandoci in un vertice degenere, l'algoritmo del simplesso può generare la seguente sequenza di basi che rappresentano tutte questo stesso vertice degenere:

$$B_t \to B_{t+1} \to \cdots \to B_{t+r-1} \to B_{t+r} = B_t.$$

Una volta tornato nella base B_t questa sequenza di basi verrà di nuovo ripetuta all'infinito senza che l'algoritmo termini. Anche se non le vedremo, esistono comunque delle regole particolari per la scelta delle variabili da far entrare e uscire di base, dette *regole anticiclaggio*, che consentono all'algoritmo di terminare in un numero finito di iterazioni anche in presenza di soluzioni di base ammissibili degeneri.

Soluzioni ottime uniche e multiple

Sia data la solita base B con la riformulazione (3.6). Nel caso in cui valga una condizione più forte rispetto alla condizione di ottimalità (4.1) e cioè se vale:

$$\gamma_j < 0 \quad j = 1, \ldots, m,$$

allora possiamo dire con certezza che la soluzione di base associata a B non solo è soluzione ottima del problema ma è anche *l'unica soluzione ottima* del problema. Infatti, per il valore dell'obiettivo in S_a si avrà:

$$\gamma_0 + \sum_{j=1}^{n-m} \underbrace{\gamma_j}_{<0} \underbrace{x_{i_{m+j}}}_{\geq 0} \leq \gamma_0,$$

ovvero in S_a il valore dell'obiettivo non può mai superare il valore γ_0 e può essere uguale a γ_0 solo se tutte le variabili $x_{i_{m+j}}$, $j = 1, \ldots, n-m$, hanno valore nullo e cioè in corrispondenza della nostra soluzione di base. Quindi tale soluzione di base è la sola soluzione ottima.
Ma cosa succede se esiste un qualche $\gamma_h = 0$? Non possiamo concludere immediatamente che esistono più soluzioni ottime. Esistono diversi casi possibili che

ci apprestiamo a descrivere. Prima però riscriviamo la riformulazione rispetto alla base B tenendo a 0 tutte le variabili fuori base tranne la $x_{i_{m+h}}$ con $\gamma_h = 0$. Avremo:

$$\begin{array}{ll} \max & \gamma_0 \\ & x_{i_1} = \beta_1 + \alpha_{1h} x_{i_{m+h}} \\ & \cdots \\ & x_{i_m} = \beta_m + \alpha_{mh} x_{i_{m+h}} \\ & x_1, \ldots, x_n \geq 0 \end{array} \tag{4.7}$$

Vediamo ora quali sono i casi possibili.

Caso 1 Se esiste h tale che $\gamma_h = 0$ e

$$\alpha_{rh} \geq 0 \quad r = 1, \ldots, m,$$

allora esiste certamente un insieme illimitato di soluzioni ottime. Infatti, se in (4.7) facciamo crescere all'infinito $x_{i_{m+h}}$, vediamo che il valore dell'obiettivo resta quello ottimo γ_0, mentre le variabili x_{i_r}, $r = 1, \ldots, m$, continuano a mantenersi non negative e quindi non si esce da S_a.

Caso 2 Se esiste h tale che $\gamma_h = 0$ e

$$\forall \;\; r: \;\; \alpha_{rh} < 0 \quad \text{si ha che} \quad \beta_r > 0,$$

allora esiste certamente un insieme limitato di soluzioni ottime. Infatti, in (4.7) possiamo far crescere $x_{i_{m+h}}$ fino al valore positivo

$$-\frac{\beta_k}{\alpha_{kh}} = \min_{r \,:\, \alpha_{rh} < 0} \left\{ -\frac{\beta_r}{\alpha_{rh}} \right\}$$

mantenendo il valore dell'obiettivo pari a quello ottimo γ_0. Quindi, un'operazione di cardine che scambi la variabile x_{i_k} con la variabile $x_{i_{m+h}}$ conduce in questo caso ad una nuova soluzione di base ammissibile ed anch'essa ottima. Risulteranno ottimi anche tutti i punti lungo il segmento che congiunge le due soluzioni di base ammissibili (o vertici) ottime.

Caso 3 Se per ogni h tale che $\gamma_h = 0$ si ha che:

$$\exists \;\; r: \;\; \alpha_{rh} < 0 \quad \text{e} \quad \beta_r = 0,$$

allora non possiamo dire se esiste un'unica soluzione ottima o se vi sono soluzioni ottime multiple senza fare ulteriori considerazioni. In questo caso infatti, per poter restare in S_a in (4.7) possiamo solo mantenere il valore di $x_{i_{m+h}}$ pari a 0 e quindi rimanere nella soluzione di base corrente.

I diversi casi saranno ora illustrati attraverso alcuni esempi.

Esempio 21 *Sia data la riformulazione rispetto alla base* $\{x_3, x_4\}$ *di un problema di PL.*

$$\begin{aligned} \max \quad & 4 - x_1 - x_2 \\ & x_3 = 2 + x_1 - x_2 \\ & x_4 = 1 - 2x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

In questo caso tutti i coefficienti di costo ridotto sono strettamente negativi, quindi la soluzione di base è ottima ed è l'unica soluzione ottima.
Sia data la riformulazione rispetto alla base $\{x_3, x_4\}$ *di un problema di PL.*

$$\begin{aligned} \max \quad & 4 - x_2 \\ & x_3 = 2 + x_1 - x_2 \\ & x_4 = 1 - 2x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Qui ci troviamo nel Caso 1: il coefficiente di costo ridotto di x_1 *è nullo e i coefficienti di* x_1 *nelle equazioni sono tutti non negativi. Quindi tutti i punti del seguente insieme:*

$$(t, 0, 2 + t, 1) \quad \forall\ t \geq 0,$$

sono soluzioni ottime del problema (si noti che $t = 0$ *coincide con la soluzione di base associata a* $\{x_3, x_4\}$*).*
Sia data la riformulazione rispetto alla base $\{x_3, x_4\}$ *di un problema di PL.*

$$\begin{aligned} \max \quad & 4 - x_2 \\ & x_3 = 2 + x_1 - x_2 \\ & x_4 = 1 - x_1 + 2x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Qui ci troviamo nel Caso 2: il coefficiente di costo ridotto di x_1 *è nullo e nelle equazioni in cui il coefficiente di* x_1 *è negativo, il termine noto è positivo. Quindi tutti i punti del seguente insieme:*

$$(t, 0, 2 + t, 1 - t) \quad 0 \leq t \leq 1,$$

sono soluzioni ottime del problema (si noti che $t = 0$ *coincide con la soluzione di base associata a* $\{x_3, x_4\}$ *e* $t = 1$ *con quella adiacente* $\{x_1, x_3\}$*).*
Sia data la riformulazione rispetto alla base $\{x_3, x_4\}$ *di un problema di PL.*

$$\begin{aligned} \max \quad & 4 - x_2 \\ & x_3 = 2 + x_1 - x_2 \\ & x_4 = -x_1 - x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Qui ci troviamo nel Caso 3: il coefficiente di costo ridotto di x_1 è nullo e in una equazione in cui il coefficiente di x_1 è negativo, il termine noto è nullo. In questo particolare esempio esiste una sola soluzione ottima. Più precisamente esiste una sola soluzione ammissibile. Infatti, l'equazione

$$x_4 = -x_1 - x_2$$

può essere soddisfatta in S_a solo se le variabili x_1 e x_2 sono entrambe nulle, ovvero in corrispondenza della soluzione di base associata a $\{x_3, x_4\}$.
Sia data la riformulazione rispetto alla base $\{x_4, x_5\}$ di un problema di PL.

$$\begin{aligned} \max \quad & 2 - x_3 \\ & x_4 = x_1 - x_2 - x_3 \\ & x_5 = -x_1 + x_2 + x_3 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Qui ci troviamo ancora nel Caso 3: il coefficiente di costo ridotto di x_1 e x_2 è nullo, in almeno una equazione in cui il coefficiente di x_1 è negativo si ha che il termine noto è nullo e lo stesso vale per x_2. Si può dimostrare però che in questo caso tutte le soluzioni:

$$x_3 = x_4 = x_5 = 0 \quad x_1 = x_2 = \alpha \ \ \forall\ \alpha \geq 0,$$

sono ammissibili e ottime (il caso $\alpha = 0$ coincide con la soluzione di base associata alla base $\{x_4, x_5\}$).

Come individuare la matrice $\mathbf{A}_B^{-1}$

Come vedremo è importante conoscere, data una base B e la relativa matrice $\mathbf{A}_B$, l'inversa $\mathbf{A}_B^{-1}$ di tale matrice. Non è però sempre necessario calcolare da zero tale inversa. In alcuni casi la riformulazione rispetto alla base B ci fornisce già la matrice $\mathbf{A}_B^{-1}$.
Supponiamo che alcune colonne della matrice $\mathbf{A}$, la matrice originale dei vincoli di uguaglianza del problema, formino la matrice identica $\mathbf{I}$, ovvero che esistano m variabili $[x_{t_1}, \ldots, x_{t_m}]$ le cui corrispondenti colonne nella matrice $\mathbf{A}$ formino la matrice identica di ordine $m \times m$. Queste variabili rappresentano una base per il problema di PL (la matrice identica è ovviamente invertibile) e in particolare una base ammissibile se tutti i termini noti delle equazioni sono non negativi. Vediamo un esempio di ciò.

Esempio 22 *Sia dato il problema di PL:*

$$\begin{aligned} \max \quad & x_1 - x_3 - 2x_4 - 2x_5 \\ & x_1 + x_2 + x_4 = 8 \\ & x_1 - x_2 + x_3 = 4 \\ & x_1 + 2x_2 + x_5 = 12 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Se prendiamo le $m = 3$ variabili $[x_4, x_3, x_5]$, possiamo vedere che le colonne corrispondenti:

$$x_4 \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad x_3 \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad x_5 \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

formano la matrice identica di ordine 3×3 e che queste 3 variabili formano una base (in questo caso ammissibile) per il nostro problema.

Supponiamo ora di essere arrivati tramite una serie di operazioni di cardine alla riformulazione rispetto alla base $B = \{x_{i_1}, \ldots, x_{i_m}\}$ del problema di PL:

$$\begin{aligned} \max \quad & \mathbf{c}_B \mathbf{A}_B^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B \mathbf{A}_B^{-1}\mathbf{A}_N)\mathbf{x}_N \\ & \mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N \\ & \mathbf{x}_B, \mathbf{x}_N \geq 0 \end{aligned}$$

Riscriviamo questa portando tutte le variabili fuori base nella parte sinistra delle equazioni dei vincoli, ovvero:

$$\begin{aligned} \max \quad & \mathbf{c}_B \mathbf{A}_B^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B \mathbf{A}_B^{-1}\mathbf{A}_N)\mathbf{x}_N \\ & \mathbf{x}_B + \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N = \mathbf{A}_B^{-1}\mathbf{b} \\ & \mathbf{x}_B, \mathbf{x}_N \geq 0 \end{aligned} \tag{4.8}$$

A questo punto l'inversa $\mathbf{A}_B^{-1}$ si legge in (4.8) nel modo seguente: la prima colonna di $\mathbf{A}_B^{-1}$ è la colonna relativa a x_{t_1} in (4.8), la seconda colonna di $\mathbf{A}_B^{-1}$ è la colonna relativa a x_{t_2} in (4.8), eccetera fino alla m-esima colonna di $\mathbf{A}_B^{-1}$ che è la colonna relativa a x_{t_m} in (4.8) (si noti che l'ordine delle variabili x_{t_r}, $r = 1, \ldots, m$ è qui essenziale). Ma vediamo di chiarire meglio la cosa attraverso il nostro esempio.

Esempio 23 *Possiamo partire proprio dalla base $\{x_4, x_3, x_5\}$ che risulta ammissibile. La riformulazione rispetto a questa base è la seguente:*

$$\begin{aligned} \max \quad & -44 + 6x_1 + 3x_2 \\ & x_4 = 8 - x_1 - x_2 \\ & x_3 = 4 - x_1 + x_2 \\ & x_5 = 12 - x_1 - 2x_2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Con una prima operazione di cardine scambiamo x_1 e x_3 nella base. La riformulazione rispetto alla nuova base $\{x_4, x_1, x_5\}$ è la seguente:

$$\begin{aligned} \max \quad & -24 - 6x_3 + 9x_2 \\ & x_4 = 4 + x_3 - 2x_2 \\ & x_1 = 4 - x_3 + x_2 \\ & x_5 = 8 + x_3 - 3x_2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Poi, con una seconda operazione di cardine scambiamo x_2 e x_4 nella base. La riformulazione rispetto alla nuova base $\{x_2, x_1, x_5\}$ è la seguente:

$$\begin{aligned} \max \quad & -6 - 3/2x_3 - 9/2x_4 \\ & x_2 = 2 + 1/2x_3 - 1/2x_4 \\ & x_1 = 6 - 1/2x_3 - 1/2x_4 \\ & x_5 = 2 - 1/2x_3 + 3/2x_4 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

A questo punto ci chiediamo: data la base $B = \{x_2, x_1, x_5\}$ con la relativa matrice:

$$\mathbf{A}_B = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}$$

qual è l'inversa di tale matrice? Come detto, questa si può ottenere osservando la riformulazione rispetto alla base B in cui però nelle equazioni spostiamo tutti i termini relativi alle variabili fuori base a sinistra, cioè nel nostro esempio:

$$\begin{aligned} \max \quad & -6 - 3/2x_3 - 9/2x_4 \\ & x_2 - 1/2x_3 + 1/2x_4 = 2 \\ & x_1 + 1/2x_3 + 1/2x_4 = 6 \\ & x_5 + 1/2x_3 - 3/2x_4 = 2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Si avrà che la prima colonna di $\mathbf{A}_B^{-1}$ è la colonna della variabile x_4 in quest'ultima formulazione, ovvero

$$\begin{pmatrix} 1/2 \\ 1/2 \\ -3/2 \end{pmatrix}$$

la seconda quella relativa a x_3:

$$\begin{pmatrix} -1/2 \\ 1/2 \\ 1/2 \end{pmatrix}$$

e la terza quella relativa a x_5:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Avremo quindi:

$$\mathbf{A}_B^{-1} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ 1/2 & 1/2 & 0 \\ -3/2 & 1/2 & 1 \end{bmatrix}$$

4.2 Il metodo due fasi

Descriveremo ora un metodo, detto *metodo due fasi*, che, dato un problema di PL, ci consente di stabilire se $S_a = \emptyset$ o, in caso contrario, ci restituisce una base ammissibile del problema. Ricordiamo che questo è essenziale perché l'algoritmo del simplesso può partire solo se si ha a disposizione una base ammissibile.
Sia dato il problema di PL in forma standard:

$$\begin{array}{ll} \max & cx \\ & a_i x = b_i \quad i = 1, \ldots, m \\ & x_j \geq 0 \quad\;\; j = 1, \ldots, n \end{array}$$

Chiameremo questo problema *problema di II fase*. A esso associamo il seguente problema, detto *problema di I fase*:

$$\begin{array}{lll} \xi^* = \max & -\sum_{i=1}^m s_i & \\ & a_i x + s_i = b_i & i \in \{1, \ldots, m\} : \; b_i \geq 0 \\ & a_i x - s_i = b_i & i \in \{1, \ldots, m\} : \; b_i < 0 \\ & x_j \geq 0 & j = 1, \ldots, n \\ & s_i \geq 0 & i = 1, \ldots, m \end{array}$$

Quindi aggiungiamo o sottraiamo (a seconda del segno del termine noto) una variabile non negativa s_i in ogni vincolo del problema originario e l'obiettivo del problema è la somma, cambiata di segno, di tali variabili. Per prima cosa notiamo che $s_i \geq 0$, $i = 1, \ldots, m$, implica che

$$-\sum_{i=1}^{m} s_i \leq 0$$

e quindi l'obiettivo del problema di I fase non può essere illimitato. Inoltre, la soluzione di base associata alla base $\{s_1, \ldots, s_m\}$ è ammissibile. Infatti, si ha la seguente riformulazione del problema di I fase rispetto a questa base:

$$\begin{array}{lll} \xi^* = \max & -\sum_{i:\; b_i \geq 0}(b_i - a_i x) - \sum_{i:\; b_i < 0}(-b_i + a_i x) & \\ & s_i = b_i - a_i x & i \in \{1, \ldots, m\} : \; b_i \geq 0 \\ & s_i = -b_i + a_i x & i \in \{1, \ldots, m\} : \; b_i < 0 \\ & x_j \geq 0 & j = 1, \ldots, n \\ & s_i \geq 0 & i = 1, \ldots, m \end{array}$$

da cui si ottiene la soluzione di base:

$$s_i = b_i \;\; i : \; b_i \geq 0, \quad s_i = -b_i \;\; i : \; b_i < 0, \quad x_j = 0 \;\; j = 1, \ldots, n,$$

che è ammissibile. Quindi il problema di I fase ha regione ammissibile non vuota e obiettivo non illimitato. Ne consegue che esso ammette soluzione ottima e valore ottimo sicuramente non superiore a 0. Vale la seguente osservazione.

Osservazione 14 *Il problema di I fase ha valore ottimo ξ^* pari a 0 se e solo se il problema di II fase ha regione ammissibile S_a non vuota.*

Dimostrazione Supponiamo dapprima che $\xi^* = 0$ e dimostriamo che $S_a \neq \emptyset$. Ma $\xi^* = 0$ vuol dire che esiste una soluzione del problema di I fase che indichiamo con $(\overline{s}, \overline{x})$ con tutte le variabili $\overline{s}_i = 0$, cioè $\overline{s} = 0$. Se sostituiamo questa soluzione nei vincoli del problema di I fase otteniamo:

$$\begin{aligned} \overline{s}_i &= b_i - a_i\overline{x} \quad & i \in \{1,\dots,m\} :\ b_i \geq 0 \\ \overline{s}_i &= -b_i + a_i\overline{x} \quad & i \in \{1,\dots,m\} :\ b_i < 0 \\ \overline{x}_j &\geq 0 & j = 1,\dots,n \\ \overline{s}_i &= 0 & i = 1,\dots,m \end{aligned}$$

o, equivalentemente:

$$a_i\overline{x} = b_i \quad i = 1,\dots,m \quad \overline{x} \geq 0,$$

da cui si ricava che $\overline{x} \in S_a$.
Supponiamo ora invece che $S_a \neq \emptyset$ e dimostriamo che questo implica che $\xi^* = 0$. Dato $\overline{x} \in S_a$, si verifica facilmente che la soluzione $(\overline{s}, \overline{x})$ con

$$\overline{s}_i = 0 \quad i = 1,\dots,m,$$

è ammissibile per il problema di I fase e ha valore dell'obiettivo:

$$-\sum_{i=1}^{m} \overline{s}_i = 0.$$

Poiché, come già osservato, il valore dell'obiettivo del problema di I fase non può essere superiore a 0, questa soluzione ammissibile è anche ottima per il problema di I fase e il valore ottimo ξ^* è pari a 0 come si voleva dimostrare.

Risolviamo ora il problema di I fase utilizzando il metodo del simplesso. Si noti che qui *abbiamo già a disposizione una base ammissibile* (la base $\{s_1,\dots,s_m\}$). Avremo le seguenti due possibilità per il valore ottimo ξ^* del problema:

$\xi^* < 0$ Allora, in base all'Osservazione 14, possiamo concludere che $S_a = \emptyset$, cioè la regione ammissibile del problema di II fase è vuota.

$\xi^* = 0$ Allora $S_a \neq \emptyset$ (la soluzione ottima del problema di I fase è già una soluzione ammissibile per il problema di II fase) e abbiamo due casi possibili:

Caso 1 tutte le variabili s_i sono al di fuori della base ottima del problema di I fase. In tal caso la base ottima del problema di I fase è già una base ammissibile del problema di II fase. La riformulazione del problema di II fase rispetto a questa base si può ottenere semplicemente dalla riformulazione del problema di I fase rispetto a questa base, eliminando da quest'ultima tutte le variabili s_i e sostituendo l'obiettivo del problema di I fase con quello del problema di II fase (si vedano gli esempi successivi per chiarire meglio il procedimento).

Caso 2 Alcune variabili s_i sono nella base ottima del problema di I fase. In tal caso si operano, fino a quando è possibile, delle operazioni di cardine per far uscire di base le variabili s_i attualmente in base facendo entrare al loro posto solo variabili x_j. Se si riesce a far uscire dalla base tutte le variabili s_i ci si ritrova infine nella stessa situazione del Caso 1 e si procede nello stesso modo. Può però succedere che non si riescano a far uscire di base alcune variabili s_i. Questo, come vedremo, accade quando vi sono vincoli ridondanti, ovvero vincoli ottenibili come combinazioni lineari di altri e che quindi possono essere semplicemente eliminati. Se ci si ritrova in questa situazione, possiamo eliminare le variabili s_i in base e i relativi vincoli. Una volta proceduto a tale eliminazione ci si ritrova nella stessa situazione del Caso 1 e si procede come in quel caso (anche qui si vedano gli esempi successivi per chiarire meglio il procedimento).

Una semplificazione che si può fare nella costruzione del problema di I fase è la seguente. Se una variabile x_j compare in un solo vincolo con coefficiente dello stesso segno del termine noto, allora in quel vincolo possiamo non introdurre una variabile s. In tal caso la variabile x_j farà parte della base ammissibile iniziale per il problema di I fase al posto della variabile s non inserita. Un caso estremo di ciò si ha quando in *ogni* vincolo abbiamo una variabile x_j che compare solo in quel vincolo e con coefficiente dello stesso segno del termine noto: in tal caso non abbiamo neppure bisogno del problema di I fase, perché l'insieme di tali variabili rappresenta già una base ammissibile per il problema di II fase.

Vediamo ora alcuni esempi che serviranno a chiarire quanto visto sopra.

Esempio 24 *Si consideri il seguente problema di PL.*

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ & -x_1 - x_2 + x_3 = -1 \\ & x_1 + x_2 + x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Chiameremo questo problema di II fase. Il problema di I fase associato è il seguente:

$$\begin{aligned} \max \quad & -s_1 - s_2 \\ & -x_1 - x_2 + x_3 - s_1 = -1 \\ & x_1 + x_2 + x_4 + s_2 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1, s_2 \geq 0 \end{aligned}$$

che riformulato rispetto alla base $\{s_1, s_2\}$ diventa:

$$\begin{aligned} \max \quad & -3 + 2x_1 + 2x_2 - x_3 + x_4 \\ & s_1 = 1 - x_1 - x_2 + x_3 \\ & s_2 = 2 - x_1 - x_2 - x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1, s_2 \geq 0. \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_1 e farne uscire s_1. La riformulazione rispetto alla nuova base $\{x_1, s_2\}$ è la seguente:

$$\begin{aligned} \max \quad & -1 - 2s_1 + x_3 + x_4 \\ & x_1 = 1 - s_1 - x_2 + x_3 \\ & s_2 = 1 + s_1 - x_3 - x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1, s_2 \geq 0. \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_3 e farne uscire s_2. La riformulazione rispetto alla nuova base $\{x_1, x_3\}$ è la seguente:

$$\begin{aligned} \max \quad & -s_1 - s_2 \\ & x_1 = 2 - x_2 - s_2 - x_4 \\ & x_3 = 1 + s_1 - s_2 - x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1, s_2 \geq 0. \end{aligned} \tag{4.9}$$

A questo punto abbiamo una soluzione ottima per il problema di I fase con valore ottimo $\xi^* = 0$. Quindi il problema di II fase ha regione ammissibile non vuota. Inoltre, ci troviamo nel Caso 1, dal momento che tutte le variabili s_i sono fuori dalla base ottima $\{x_1, x_3\}$. Possiamo quindi usare questa base come base ammissibile iniziale per il problema di II fase. La riformulazione rispetto a questa base si ottiene dalla riformulazione (4.9) da cui si scartano tutte le variabili s_i e si sostituisce l'obiettivo con quello di II fase

$$x_1 + 2x_2$$

dove le variabili in base (in questa caso la x_1) dovranno essere sostituite dalle espressioni date dalle relative equazioni (qui $x_1 = 2 - x_2 - x_4$). Avremo quindi nell'esempio la seguente riformulazione:

$$\begin{aligned} \max \quad & 2 + x_2 - x_4 \\ & x_1 = 2 - x_2 - x_4 \\ & x_3 = 1 - x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

A questo punto si può procedere con il metodo del simplesso per risolvere il problema di II fase.

Teniamo presente che in questo esempio avremmo potuto evitare di introdurre una variabile s. Più precisamente osserviamo che la varibaile x_4 compare solo nel secondo vincolo e ha coefficiente con lo stesso segno del termine noto (anche la x_3 compare solo nel primo vincolo ma il suo coefficiente ha segno opposto rispetto al termine noto). Quindi posssiamo semplificare il problema di I fase senza introdurre la varibaile s_2:

$$\begin{aligned} \max \quad & -s_1 \\ & -x_1 - x_2 + x_3 - s_1 = -1 \\ & x_1 + x_2 + x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1 \geq 0 \end{aligned}$$

con la base ammissibile iniziale $\{s_1, x_4\}$ per esso e la seguente riformulazione rispetto a essa:

$$\begin{aligned} \max \quad & -1 + x_1 + x_2 - x_3 \\ & s_1 = 1 - x_1 - x_2 + x_3 \\ & x_4 = 2 - x_1 - x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1 \geq 0. \end{aligned}$$

Si consideri il seguente problema di PL.

$$\begin{aligned} \max \quad & 2x_1 + x_2 + 1/2x_3 \\ & x_1 + x_2 + x_3 = 3 \\ & x_1 - x_2 + x_3 = 1 \\ & x_1 + x_3 = 2 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Al solito, chiameremo questo problema di II fase. Il problema di I fase associato è il seguente:

$$\begin{aligned} \max \quad & -s_1 - s_2 - s_3 \\ & x_1 + x_2 + x_3 + s_1 = 3 \\ & x_1 - x_2 + x_3 + s_2 = 1 \\ & x_1 + x_3 + s_3 = 2 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2, s_3 \geq 0 \end{aligned}$$

che riformulato rispetto alla base $\{s_1, s_2, s_3\}$ diventa:

$$\begin{aligned} \max \quad & -6 + 3x_1 + 3x_3 \\ & s_1 = 3 - x_1 - x_2 - x_3 \\ & s_2 = 1 - x_1 + x_2 - x_3 \\ & s_3 = 2 - x_1 - x_3 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2, s_3 \geq 0. \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_1 e farne uscire s_2. La riformulazione rispetto alla nuova base $\{s_1, x_1, s_3\}$ è la seguente:

$$\begin{aligned} \max \quad & -3 - 3s_2 + 3x_2 \\ & s_1 = 2 + s_2 - 2x_2 \\ & x_1 = 1 - s_2 + x_2 - x_3 \\ & s_3 = 1 + s_2 - x_2 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2, s_3 \geq 0. \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_2 e farne uscire s_1. La riformulazione rispetto alla nuova base $\{s_1, x_1, s_3\}$ è la seguente:

$$\begin{aligned} \max \quad & -3/2s_2 - 3/2s_1 \\ & x_2 = 1 + 1/2s_2 - 1/2s_1 \\ & x_1 = 2 - 1/2s_2 - 1/2s_1 - x_3 \\ & s_3 = 1/2s_2 + 1/2s_1 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2, s_3 \geq 0. \end{aligned} \tag{4.10}$$

Abbiamo una soluzione ottima del problema di I fase e si ha $\xi^* = 0$. Quindi il problema di II fase ha regione ammissibile non vuota. Ci troviamo però ora nel Caso 2, con una variabile (la s_3) che fa parte della base ottima $\{x_1, x_2, s_3\}$. Per prima cosa dobbiamo tentare di far uscire s_3 di base facendo entrare una variabile x_i fuori base. Ma l'unica possibile candidata, la variabile x_3, ha coefficiente nullo (e quindi non appare) nell'equazione realtiva a s_3, cioè

$$s_3 = 1/2s_2 + 1/2s_1.$$

La stessa equazione ci dice anche che il vincolo relativo alla variabile s_3 (cioè $x_1+x_3 = 2$) è ottenibile come combinazione lineare con coefficienti entrambi pari a $1/2$ dei vincoli relativi alle variabili s_1 ($x_1+x_2+x_3 = 3$) e s_2 ($x_1-x_2+x_3 = 1$). Ne consegue che questo vincolo è ridondante e possiamo scartarlo insieme alla relativa variabile s_3. Una volta soppressa la variabile s_3 e l'equazione ad essa relativa, la riformulazione (4.10) diventa:

$$\begin{aligned} \max \quad & -3/2s_2 - 3/2s_1 \\ & x_2 = 1 + 1/2s_2 - 1/2s_1 \\ & x_1 = 2 - 1/2s_2 - 1/2s_1 - x_3 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2 \geq 0. \end{aligned} \tag{4.11}$$

A questo punto abbiamo la base ottima $\{x_1, x_2\}$ per il problema di I fase che non contiene variabili s_i. Questa base è ammissibile per il problema di II fase e la riformulazione rispetto ad essa si ottiene sopprimendo in (4.11) le variabili s_i e sostituendo l'obiettivo con quello di II fase

$$2x_1 + x_2 + 1/2x_3$$

con la sostituzione delle variabili in base con le relative espressione ottenute dalle equazioni $x_2 = 1$ e $x_1 = 2 - x_3$. Si arriva quindi alla riformulazione rispetto alla base $\{x_2, x_1\}$:

$$\begin{aligned} \max \quad & 5 - 3/2x_3 \\ & x_2 = 1 \\ & x_1 = 2 - x_3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{4.12}$$

Possiamo quindi risolvere il problema di II fase utilizzando il metodo del simplesso.

Si consideri il seguente problema di PL.

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 + x_2 = 3 \\ & x_1 + x_3 = 1 \\ & x_2 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Al solito, chiameremo questo problema di II fase. Il problema di I fase associato è il seguente:

$$\begin{aligned} \max \quad & -s_1 - s_2 - s_3 \\ & x_1 + x_2 + s_1 = 3 \\ & x_1 + x_3 + s_2 = 1 \\ & x_2 + x_4 + s_3 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \quad s_1, s_2, s_3 \geq 0 \end{aligned}$$

che riformulato rispetto alla base $\{s_1, s_2, s_3\}$ diventa:

$$\begin{aligned} \max \quad & -5 + 2x_1 + 2x_2 + x_3 + x_4 \\ & s_1 = 3 - x_1 - x_2 \\ & s_2 = 1 - x_1 - x_3 \\ & s_3 = 1 - x_2 - x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \qquad s_1, s_2, s_3 \geq 0 \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_1 e farne uscire s_2. La riformulazione rispetto alla nuova base $\{s_1, x_1, s_3\}$ è la seguente:

$$\begin{aligned} \max \quad & -3 - 2s_2 + 2x_2 - x_3 + x_4 \\ & s_1 = 2 - 3s_2 - x_3 - x_2 \\ & x_1 = 1 - s_2 - x_3 \\ & s_3 = 1 - x_2 - x_4 \\ & x_1, x_2, x_3 \geq 0 \qquad s_1, s_2, s_3 \geq 0. \end{aligned}$$

In base alle regole viste per il metodo del simplesso, dovremo far entrare in base x_2 e farne uscire s_3. La riformulazione rispetto alla nuova base $\{s_1, x_1, x_2\}$ è la seguente:

$$\begin{aligned} \max \quad & -1 - 2s_2 - 2s_3 - x_3 - x_4 \\ & s_1 = 1 - 3s_2 - x_3 + s_3 + x_4 \\ & x_1 = 1 - s_2 - x_3 \\ & x_2 = 1 - s_3 - x_4 \\ & x_1, x_2, x_3 \geq 0 \qquad\qquad s_1, s_2, s_3 \geq 0. \end{aligned}$$

Abbiamo una soluzione ottima del problema di I fase e si ha $\xi^* = -1 < 0$. Possiamo quindi immediatamente concludere che il problema di II fase ha regione ammissibile vuota.

Capitolo 5

Dualità

Dato un problema di PL in forma standard

$$\begin{aligned} \max \quad & \mathbf{cx} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{5.1}$$

che chiameremo problema *primale*, possiamo associare a esso un altro problema di PL, detto problema *duale*, definito come segue ($\mathbf{u}$ è il vettore di variabili duali con m componenti)

$$\begin{aligned} \min \quad & \mathbf{ub} \\ & \mathbf{uA} \geq \mathbf{c} \end{aligned} \tag{5.2}$$

o, in forma scalare:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{m} u_i b_i \\ & \textstyle\sum_{i=1}^{m} u_i a_{ij} \geq c_j \quad j = 1, \ldots, n \end{aligned} \tag{5.3}$$

Indichiamo con

$$D_a = \{\mathbf{u} \in R^m \ : \ \mathbf{uA} \geq \mathbf{c}\}$$

la regione ammissibile del problema duale e con

$$D_{ott} = \{\mathbf{u}^* \in D_a \ : \ \mathbf{u}^*\mathbf{b} \leq \mathbf{ub} \ \ \forall \ \mathbf{u} \in D_a\}$$

l'insieme delle sue soluzioni ottime. Si può notare che esiste una stretta relazione tra

- variabili del primale e vincoli del duale;
- vincoli del primale e variabili del duale.

In particolare notiamo che

1. nel primale ci sono n variabili esattamente come nel duale vi sono n vincoli. Inoltre, i coefficienti del j-esimo vincolo del duale coincidono con i coefficienti della variabile x_j nei vincoli del primale, mentre il termine noto del j-esimo vincolo del duale coincide con il coefficiente di x_j nell'obiettivo del primale.

2. Nel primale vi sono m vincoli esattamente come nel duale vi sono m variabili. Inoltre, i coefficienti dell'i-esima variabile u_i del duale coincidono con i coefficienti dell'i-esimo vincolo del primale, mentre il coefficiente di u_i nell'obiettivo del duale coincide con il termine noto dell'i-esimo vincolo del primale.

Come esercizio si mostri che il duale del seguente problema di PL in forma standard:

$$\begin{array}{ll} \max & x_1 + x_2 \\ & 3x_1 + 2x_2 + x_3 = 5 \\ & 4x_1 + 5x_2 + x_4 = 4 \\ & x_2 + x_5 = 2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array}$$

è il seguente problema:

$$\begin{array}{ll} \min & 5u_1 + 4u_2 + 2u_3 \\ & 3u_1 + 4u_2 \geq 1 \\ & 2u_1 + 5u_2 + u_3 \geq 1 \\ & u_1 \geq 0 \\ & u_2 \geq 0 \\ & u_3 \geq 0. \end{array}$$

Le soluzioni dei due problemi primale e duale sono fortemente legate tra loro come dimostra una serie di risultati.

Osservazione 15 *Per ogni* $\mathbf{x}_0 \in S_a$ *e per ogni* $\mathbf{u}_0 \in D_a$ *si ha che*

$$\mathbf{c}\mathbf{x}_0 \leq \mathbf{u}_0\mathbf{b}.$$

Dimostrazione $\mathbf{x}_0 \in S_a$ implica

$$\mathbf{A}\mathbf{x}_0 = \mathbf{b} \tag{5.4}$$

Moltiplicando entrambi i membri di (5.4) per $\mathbf{u}_0$ si ottiene

$$\mathbf{u}_0\mathbf{b} = \mathbf{u}_0\mathbf{A}\mathbf{x}_0 = (\mathbf{u}_0\mathbf{A})\mathbf{x}_0. \tag{5.5}$$

Ma $\mathbf{u}_0 \in D_a$ implica

$$\mathbf{u}_0\mathbf{A} \geq \mathbf{c}$$

da cui, moltiplicando entrambi i membri per $\mathbf{x}_0$ (che è ≥ 0 per l'appartenenza di $\mathbf{x}_0$ a S_a), si ottiene

$$(\mathbf{u}_0\mathbf{A})\mathbf{x}_0 \geq \mathbf{c}\mathbf{x}_0$$

che, combinato con (5.5), dimostra il risultato.

Osservazione 16 *Se* $\mathbf{x}^* \in S_a$ *e* $\mathbf{u}^* \in D_a$ *e inoltre*

$$\mathbf{c}\mathbf{x}^* = \mathbf{u}^*\mathbf{b}$$

allora $\mathbf{x}^* \in S_{ott}$ *e* $\mathbf{u}^* \in D_{ott}$.

Dimostrazione In base all'Osservazione 15 si ha che

$$\forall\ \mathbf{x} \in S_a \qquad \mathbf{c}\mathbf{x} \leq \mathbf{u}^*\mathbf{b}.$$

Ma essendo $\mathbf{c}\mathbf{x}^* = \mathbf{u}^*\mathbf{b}$ si ha anche

$$\forall\ \mathbf{x} \in S_a \qquad \mathbf{c}\mathbf{x} \leq \mathbf{c}\mathbf{x}^*$$

il che equivale a dire che $\mathbf{x}^* \in S_{ott}$. In modo del tutto analogo si dimostra che $\mathbf{u}^* \in D_{ott}$.

Osservazione 17 *Se uno dei due problemi ha obiettivo illimitato, allora l'altro ha regione ammissibile vuota.*

Dimostrazione Dimostriamo che se l'obiettivo primale è illimitato, allora $D_a = \emptyset$ (la dimostrazione che l'obiettivo duale illimitato implica $S_a = \emptyset$ è del tutto analoga). Supponiamo per assurdo che $D_a \neq \emptyset$ e sia $\mathbf{u}_0 \in D_a$. In base all'Osservazione 15 si ha che

$$\forall\ \mathbf{x} \in S_a \qquad \mathbf{c}\mathbf{x} \leq \mathbf{u}_0\mathbf{b}$$

e quindi l'obiettivo del primale è limitato dal valore $\mathbf{u}_0\mathbf{b}$, il che contraddice l'illimitatezza di tale obiettivo.

Data una base B sappiamo che a questa è associata la soluzione di base del problema primale:

$$\mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b} \quad \mathbf{x}_N = 0.$$

Ma a questa base possiamo associare anche la seguente soluzione di base del duale:

$$\mathbf{u}^B = \mathbf{c}_B\mathbf{A}_B^{-1}.$$

Vediamo quando questa soluzione di base del duale è ammissibile per il duale. Per essere ammissibile deve soddisfare i vincoli:

$$\mathbf{u}^B\mathbf{A} \geq \mathbf{c}$$

o, equivalentemente, separando i vincoli duali associati alle variabili nella base B da quelli associati alle variabili fuori base, ovvero in N:

$$\mathbf{u}^B\mathbf{A}_B \geq \mathbf{c}_B \tag{5.6}$$
$$\mathbf{u}^B\mathbf{A}_N \geq \mathbf{c}_N \tag{5.7}$$

Ma:

$$\mathbf{u}^B\mathbf{A}_B = \mathbf{c}_B\mathbf{A}_B^{-1}\mathbf{A}_B = \mathbf{c}_B,$$

e quindi i vincoli (5.6) sono certamente soddisfatti, mentre

$$\mathbf{u}^B\mathbf{A}_N = \mathbf{c}_B\mathbf{A}_B^{-1}\mathbf{A}_N,$$

e quindi i vincoli (5.7) sono equivalenti a:

$$\mathbf{c}_N - \mathbf{c}_B\mathbf{A}_B^{-1}\mathbf{A}_N \leq 0$$

(si noti la coincidenza con la condizione di ottimalità (4.2) per il problema primale). In particolare se

$$\mathbf{c}_N - \mathbf{c}_B\mathbf{A}_B^{-1}\mathbf{A}_N < 0$$

la soluzione di base del duale $\mathbf{u}^B$ ammissibile verrà detta *non degenere*, altrimenti verrà detta *degenere*. Si noti che le due soluzioni di base rispettivamente del primale e del duale associate alla base B hanno lo stesso valore dell'obiettivo:

$$\mathbf{c}_B\mathbf{x}_B = \mathbf{c}_B\mathbf{A}_B^{-1}\mathbf{b} = \mathbf{u}^B\mathbf{b},$$

e quindi, in base all'Osservazione 16, se entrambe sono ammissibili per i rispettivi problemi, sono anche soluzioni ottime degli stessi problemi.

Fino a questo punto abbiamo definito solo il duale di un problema in forma standard. Ma sappiamo che ogni problema di PL può essere ricondotto alla forma standard. Tra questi anche il duale (5.2) del nostro problema primale (5.1). Quindi ci si può porre la seguente domanda: una volta ricondotto il duale (5.2) in forma standard, quale forma avrà il suo duale?

Osservazione 18 *Il duale del problema duale (5.2) coincide con il problema primale (5.1).*

Dimostrazione Trasformiamo il duale in forma standard:

$$\begin{array}{ll} -\max & -(\mathbf{u}' - \mathbf{u}'')\mathbf{b} \\ & -(\mathbf{u}' - \mathbf{u}'')\mathbf{A} + \mathbf{I}\mathbf{y} = -\mathbf{c} \\ & \mathbf{u}', \mathbf{u}'', \mathbf{y} \geq \mathbf{0} \end{array}$$

Il duale di questo problema risulta essere:

$$\begin{array}{ll} -\min & -\mathbf{c}\mathbf{z} \\ & -\mathbf{A}\mathbf{z} \geq -\mathbf{b} \\ & \mathbf{A}\mathbf{z} \geq \mathbf{b} \\ & \mathbf{z} \geq \mathbf{0} \end{array}$$

che risulta equivalente a

$$\begin{aligned} \max \quad & \mathbf{cz} \\ & \mathbf{Az} = \mathbf{b} \\ & \mathbf{z} \geq \mathbf{0} \end{aligned}$$

ovvero al problema primale.

Questa osservazione ci mostra la totale simmetria esistente tra problema primale e duale. Si noti in particolare che si è dimostrata un'implicazione del tipo:

Il primale ha la proprietà $\mathcal{P} \Rightarrow$ il duale ha la proprietà $\mathcal{P}'$

si è automaticamente dimostrato anche che

Il duale ha la proprietà $\mathcal{P} \Rightarrow$ il duale del duale ha la proprietà $\mathcal{P}'$

o, equivalentemente in base all'Osservazione 18:

Il duale ha la proprietà $\mathcal{P} \Rightarrow$ il primale ha la proprietà $\mathcal{P}'$

Per esempio, nell'Osservazione 17 abbiamo dimostrato che se il primale ha la proprietà $\mathcal{P} \equiv$ *obiettivo illimitato*, allora il duale ha la proprietà $\mathcal{P}' \equiv$ *regione ammissibile vuota*. Con questo abbiamo anche automaticamente dimostrato che se il duale ha la proprietà $\mathcal{P} \equiv$ *obiettivo illimitato*, allora il primale ha la proprietà $\mathcal{P}' \equiv$ *regione ammissibile vuota*.

Siamo ora pronti per dimostrare il *I teorema della dualità*.

Teorema 4 *Uno dei due problemi ha soluzioni ottime se e solo se anche l'altro ha soluzioni ottime. Formalmente, $S_{ott} \neq \emptyset$ se e solo se $D_{ott} \neq \emptyset$. Inoltre, i valori ottimi dei due problemi coincidono.*

Dimostrazione Supponiamo di avere $S_{ott} \neq \emptyset$ e sia B^* una base ottima per il problema primale. Avremo quindi che la soluzione di base del primale associata a B^*, cioè:

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} \quad \mathbf{x}_{N^*} = 0,$$

è ammissibile per il primale e inoltre è soddisfatta la condizione di ottimalità:

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0.$$

Ma questo ci dice anche che la soluzione di base del duale associata a B^*, cioè:

$$\mathbf{u}^{B^*} = \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1},$$

è ammissibile per il duale. Come già osservato in precedenza le due soluzioni di base del primale e del duale associate a B^* hanno lo stesso valore dell'obiettivo

ed essendo ammissibili rispettivamente per il primale e per il duale, l'Osservazione 16 ci permette di concludere che sono soluzioni ottime dei rispettivi problemi. Quindi abbiamo dimostrato che $S_{ott} \neq \emptyset$ implica $D_{ott} \neq \emptyset$ e i due valori ottimi coincidono. Il viceversa, cioè che $D_{ott} \neq \emptyset$ implica $S_{ott} \neq \emptyset$, è una conseguenza immediata della proprietà di simmetria tra primale e duale esppressa dall'Osservazione 18. Infatti, in base a quanto appena dimostrato, se il duale ha soluzioni ottime anche il suo duale ne ha e i valori ottimi dei due problemi coincidono. Ma in base all'Osservazione 18, il duale del duale coincide proprio con il problema primale.

Riassumiamo ora tutte le possibili relazioni tra primale e duale.

- In base al I teorema della dualità

$$S_{ott} \neq \emptyset \quad \Leftrightarrow \quad D_{ott} \neq \emptyset$$

  In base allo stesso teorema si ha anche che i valori ottimi coincidono.

- Se $S_{ott} = \emptyset$ in quanto l'obiettivo primale è illimitato, allora $D_a = \emptyset$ (per l'Osservazione 17). Ciò implica anche per la simmetria tra primale e duale espressa nell'Osservazione 18, che se $D_{ott} = \emptyset$ in quanto l'obiettivo duale è illimitato, allora $S_a = \emptyset$.

- Se $S_a = \emptyset$, allora $D_a = \emptyset$ oppure l'obiettivo duale è illimitato. Ciò implica anche per la simmetria tra primale e duale espressa nell'Osservazione 18, che se $D_a = \emptyset$, allora $S_a = \emptyset$ oppure l'obiettivo primale è illimitato.

Il seguente è un esempio in cui sia S_a che D_a sono insiemi vuoti.

Esempio 25 *Si consideri il seguente problema primale*

$$\begin{aligned} \max \quad & 2x_1 - x_2 \\ & x_1 - x_2 = 1 \\ & -x_1 + x_2 = -2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Si noti che $S_a = \emptyset$. Si trovi ora il duale di tale problema:

$$\begin{aligned} \min \quad & u_1 - 2u_2 \\ & u_1 - u_2 \geq 2 \\ & -u_1 + u_2 \geq -1 \end{aligned}$$

in cui si può notare che anche $D_a = \emptyset$.

Introduciamo ora il *II teorema della dualità*. Se il I teorema della dualità ci dice che i due problemi, se hanno soluzioni ottime, hanno valori ottimi coincidenti, il II teorema della dualità esprime una relazione tra le soluzioni ottime dei due problemi e, tra le altre cose, permette di ricavare le soluzioni di uno dei due problemi se è nota una soluzione dell'altro.

Teorema 5 *Si ha che* $\mathbf{x}^* \in S_{ott}$ *e* $\mathbf{u}^* \in D_{ott}$ se e solo se $\mathbf{x}^*$ *e* $\mathbf{u}^*$ *appartengono rispettivamente a* S_a *e* D_a *e soddisfano le* condizioni di complementarità, *cioè*

$$(\mathbf{u}^*\mathbf{A} - \mathbf{c})\mathbf{x}^* = 0. \tag{5.8}$$

o, in forma scalare:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} u_i^* a_{ij} - c_j \right) x_j^* = 0. \tag{5.9}$$

Dimostrazione Se vale (5.8), si ha

$$\mathbf{u}^*\mathbf{A}\mathbf{x}^* = \mathbf{c}\mathbf{x}^*.$$

Ma $\mathbf{x}^* \in S_a$ implica $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ e quindi

$$\mathbf{u}^*\mathbf{b} = \mathbf{u}^*\mathbf{A}\mathbf{x}^* = \mathbf{c}\mathbf{x}^*.$$

In base all'Osservazione 16, ciò implica che le due soluzioni sono ottime per i rispettivi problemi.
Vediamo ora di dimostrare il viceversa, ovvero che se le due soluzioni sono ottime soddisfano le condizioni di complementarità. Per il I teorema della dualità sappiamo che

$$\mathbf{u}^*\mathbf{b} = \mathbf{c}\mathbf{x}^*.$$

Ora, sommiamo e sottraiamo $\mathbf{u}^*\mathbf{A}\mathbf{x}^*$ ed otteniamo

$$\mathbf{u}^*\mathbf{b} - \mathbf{u}^*\mathbf{A}\mathbf{x}^* + \mathbf{u}^*\mathbf{A}\mathbf{x}^* - \mathbf{c}\mathbf{x}^* = 0$$

che possiamo riscrivere come segue

$$\mathbf{u}^*(\mathbf{b} - \mathbf{A}\mathbf{x}^*) + (\mathbf{u}^*\mathbf{A} - \mathbf{c})\mathbf{x}^* = 0. \tag{5.10}$$

Ora, $\mathbf{x}^* \in S_a$ implica $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ e quindi (5.10) si riduce a:

$$(\mathbf{u}^*\mathbf{A} - \mathbf{c})\mathbf{x}^* = 0,$$

che è quanto volevamo dimostrare.

Ma vediamo di capire meglio quanto ci dicono le condizioni di complementarità. Ricordiamo che

$$(\mathbf{u}^*\mathbf{A} - \mathbf{c})\mathbf{x}^* = 0,$$

in forma scalare equivale a:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} u_i^* a_{ij} - c_j \right) x_j^* = 0.$$

Dal momento che $u^* \in D_a$ e $x^* \in S_a$, si avrà che:

$$\sum_{i=1}^{m} u_i^* a_{ij} - c_j \geq 0, \quad x_j^* \geq 0 \quad \forall\ j = 1, \ldots, n.$$

e quindi

$$\left(\sum_{i=1}^{m} u_i^* a_{ij} - c_j\right) x_j^* \geq 0.$$

Ne consegue che:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} u_i^* a_{ij} - c_j\right) x_j^* = 0.$$

se e solo se:

$$\left(\sum_{i=1}^{m} u_i^* a_{ij} - c_j\right) x_j^* = 0 \quad \forall\, j = 1, \ldots, n.$$

(una somma di addendi non negativi può essere nulla se e solo se ciascun addendo è nullo). Ciò determina le seguenti implicazioni:

$$x_j^* > 0 \quad \Rightarrow \quad \sum_{i=1}^{m} u_i^* a_{ij} - c_j = 0,$$

cioè se una variabile x_j^* ha valore positivo in una soluzione ottima del problema primale, le soluzioni ottime u^* del duale devono soddisfare come uguaglianza il vincolo del duale $\sum_{i=1}^{m} u_i a_{ij} \geq c_j$ corrispondente alla variabile x_j;

$$\sum_{i=1}^{m} u_i^* a_{ij} - c_j > 0 \quad \Rightarrow \quad x_j^* = 0,$$

cioè se una soluzione ottima del duale $\mathbf{u}^*$ non soddisfa come uguaglianza il vincolo del duale $\sum_{i=1}^{m} u_i a_{ij} \geq c_j$, il valore della corrispondente variabile x_j nelle soluzioni ottime del primale deve essere nullo.
Vediamo ora come queste condizioni possono essere utilizzate per ottenere, ad esempio, una soluzione ottima del duale una volta che è data una soluzione ottima del primale. Sia dato il problema primale:

$$\begin{aligned} \max \quad & x_1 - x_2 \\ & x_1 + 2x_2 + x_3 = 6 \\ & 2x_1 + x_2 + x_4 = 4 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Si può dimostrare che la soluzione ottima di questo problema è la seguente:

$$x_1^* = 2 \; x_2^* = 0 \;\; x_3^* = 4 \;\; x_4^* = 0.$$

Si consideri ora il duale del problema:

$$\begin{aligned} \min \quad & 6u_1 + 4u_2 \\ & u_1 + 2u_2 \geq 1 \\ & 2u_1 + u_2 \geq -1 \\ & u_1 \geq 0 \\ & u_2 \geq 0 \end{aligned}$$

La condizione di complementarità è la seguente :

$$(u_1^* + 2u_2^* - 1)x_1^* + (2u_1^* + u_2^* + 1)x_2^* + (u_1^* - 0)x_3^* + (u_4^* - 0)x_4^* = 0$$

o, equivalentemente:

$$(u_1^* + 2u_2^* - 1)x_1^* = 0 \quad (2u_1^* + u_2^* + 1)x_2^* = 0 \quad (u_1^* - 0)x_3^* = 0 \quad (u_4^* - 0)x_4^* = 0$$

In base a tali condizioni si ha che:

$$x_1^* > 0 \quad \Rightarrow u_1^* + 2u_2^* = 1$$

$$x_3^* > 0 \quad \Rightarrow u_1^* = 0.$$

Quindi una soluzione ottima del duale deve soddisfare il sistema

$$\begin{gathered} u_1^* + 2u_2^* = 1 \\ u_1^* = 0 \end{gathered}$$

da cui si ricava immediatamente la soluzione ottima del duale:

$$u_1^* = 0 \quad u_2^* = 1/2.$$

Se, viceversa, supponessimo di consocere già questa soluzione ottima del duale, ma non quella del primale, allora per risalire a quest'ultima dovremmo dapprima osservare che:

$$2u_1^* + u_2^* > -1 \quad \Rightarrow \quad x_2^* = 0$$

e

$$u_2^* > 0 \quad \Rightarrow \quad x_4^* = 0.$$

Quindi, sostituendo tali valori di x_2^* e x_4^* nei vincoli di uguaglianza del primale, otteniamo il sistema

$$\begin{array}{rcl} x_1^* + x_3^* & = & 6 \\ 2x_1^* & = & 4 \end{array}$$

da cui possiamo ricavare i valori x_1^* e x_3^*.

5.1 Il simplesso duale

In precedenza abbiamo descritto il metodo del simplesso che, per distinguerlo dal metodo che descriveremo ora, chiameremo *simplesso primale*. Il metodo del simplesso primale segue il seguente schema:

- cominciamo con una base B la cui soluzione di base associata del primale è ammissibile, ma con la soluzione di base associata del duale tipicamente non ammisibile (si ricordi che l'ammissibilità dellla soluzione di base associata del duale è equivalente alla condizione di ottimalità (4.2) per il primale);

- passiamo ad altre basi con le soluzioni di base associate del primale sempre ammissibili ma con le corrispondenti soluzioni di base del duale non ammissibili;

- se esistono soluzioni ottime, ci arrestiamo quando arriviamo ad una base B^* la cui soluzione di base associata del primale è ammissibile come al solito, ma per la quale è soddisfatta la condizione di ottimalità:

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

  che equivale anche all'ammissibilità della soluzione di base associata a B^* del duale.

Quindi, nel simplesso primale si genera una successione di basi ammissibili per il primale ma non per il duale fino a raggiungere una base che sia anche ammissibile per il duale (a patto che una tale base esista , a patto cioè che il primale ammetta soluzioni ottime e non abbia obiettivo illimitato). In alcuni casi succede che è semplice avere una base la cui soluzione di base associata del duale è ammissibile ma non lo è quella del primale. Il simplesso duale non è altro che il simplesso applicato al problema duale, in cui seguiremo quindi il seguente schema:

- cominciamo con una base B la cui soluzione di base associata del duale è ammissibile, ma con la soluzione di base associata del primale tipicamente non ammisibile;

- passiamo ad altre basi con le soluzioni di base associate del duale sempre ammissibili ma con le corrispondenti soluzioni di base del primale non ammissibili;

- se esistono soluzioni ottime, ci arrestiamo quando arriviamo ad una base B^* la cui soluzione di base associata del duale è ammissibile come al solito, ma per la quale è soddisfatta anche la condizione di ammissibilità per la soluzione di base associata a B^* del primale:

$$\mathbf{A}_{B^*}^{-1}\mathbf{b} \geq 0.$$

Quindi, nel simplesso duale si genera una successione di basi ammissibili per il duale ma non per il primale fino a raggiungere una base che sia anche ammissibile per il primale (a patto che una tale base esista , a patto cioè che il duale ammetta soluzioni ottime e non abbia obiettivo illimitato). Vediamo ora di descrivere i singoli passi in un'iterazione del simplesso duale. Supporremo di avere una base B con soluzione di base associata del duale ammissibile. La riformulazione del

problema primale rispetto alla base $B = \{x_{i_1}, \ldots, x_{i_k}, \ldots, x_{i_m}\}$ è la seguente:

$$\begin{array}{ll} \max & \gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}} \\ & x_{i_1} = \beta_1 + \sum_{j=1}^{n-m} \alpha_{1j} x_{i_{m+j}} \\ & \cdots \\ & x_{i_k} = \beta_k + \sum_{j=1}^{n-m} \alpha_{kj} x_{i_{m+j}} \\ & \cdots \\ & x_{i_m} = \beta_m + \sum_{j=1}^{n-m} \alpha_{mj} x_{i_{m+j}} \\ & x_1, \ldots, x_n \geq 0 \end{array} \tag{5.11}$$

Si noti che la condizione di ammissibilità per la soluzione di base del duale è equivalente a

$$\gamma_j \leq 0 \quad j = 1, \ldots, n-m. \tag{5.12}$$

Vale la pena sottolineare che parliamo di simplesso duale ma tutte le operazioni vengono eseguite sulla *riformulazione rispetto alla base B del problema primale.*

5.1.1 Verifica di ottimalità

Se

$$\mathbf{A}_B^{-1}\mathbf{b} \geq 0,$$

o, equivalentemente:

$$\beta_i \geq 0 \quad i = 1, \ldots, m$$

allora la base B è ottima, la soluzione di base del primale

$$\mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b} \quad \mathbf{x}_N = 0$$

è ottima per il primale, mentre la soluzione di base

$$\mathbf{u}^B = \mathbf{c}_B\mathbf{A}_B^{-1}$$

è ottima per il duale.

5.1.2 Verifica di illimitatezza del duale

Se esiste un $r \in \{1, \ldots, m\}$ tale che

$$\beta_r < 0 \quad \alpha_{rj} \leq 0 \quad j = 1, \ldots, n-m,$$

allora si ha $S_a = \emptyset$. Infatti, dall'equazione:

$$x_{i_r} = \underbrace{\beta_r}_{<0} + \sum_{j=1}^{n-m} \underbrace{\alpha_{rj}}_{\leq 0} \underbrace{x_{i_{m+j}}}_{\geq 0} < 0,$$

si vede come la variabile x_{i_r} non possa mai essere non negativa se le variabili $x_{i_{m+j}}$, $j = 1, \ldots, n-m$, sono non negative. Ne consegue anche che il duale ha

obiettivo illimitato. Infatti, abbiamo visto che con $S_a = \emptyset$ sono possibili due casi: $D_a = \emptyset$ e duale con obiettivo illimitato. Ma il primo dei due casi ($D_a = \emptyset$) non si può verificare qui in quanto la soluzione di base del duale associata a B è per ipotesi ammissibile e quindi D_a non può essere vuoto.

5.1.3 Scelta della variabile uscente dalla base

Rispetto al simplesso primale qui scegliamo prima la variabile uscente dalla base. La regola di scelta è la seguente: si seleziona la variabile x_{i_k} tale che

$$\beta_k = \min\{\beta_i\} < 0$$

(per convenzione quella con indice più piccolo se il minimo è raggiunto da più variabili).

5.1.4 Scelta della variabile entrante in base

La variabile entrante in base viene scelta tra quelle con $\alpha_{kj} > 0$. In particolare si sceglie la variabile $x_{i_{m+h}}$ tale che

$$-\frac{\gamma_h}{\alpha_{kh}} = \min\left\{-\frac{\gamma_j}{\alpha_{kj}} \ : \ \alpha_{kj} > 0\right\}. \tag{5.13}$$

(nel caso il minimo sia raggiunto da più variabili si sceglie, per convenzione, quella con indice più piccolo). A questo punto possiamo eseguire l'operazione di cardine facendo uscire di base x_{i_k} e facendo entrare in base $x_{i_{m+h}}$.

5.1.5 Mantenimento dell'ammissibilità e miglioramento dell'obiettivo

Vogliamo far vedere che le scelte fatte preservano l'ammissibilità duale, cioè la soluzione di base del duale associata alla nuova base è ancora ammissibile per il duale, ed inoltre la nuova soluzione ha un valore dell'obiettivo migliore (o quantomeno non peggiore) rispetto alla precedente. Mostriamo dapprima che le scelte fatte preservano l'ammissibilità duale. Con l'operazione di cardine avremo:

$$x_{i_{m+h}} = -\frac{\beta_k}{\alpha_{kh}} + \frac{1}{\alpha_{kh}} x_{i_k} - \sum_{j=1,\ j\neq h}^{n-m} \frac{\alpha_{kj}}{\alpha_{kh}} x_{i_{m+j}},$$

da cui, sostituendo nell'obiettivo si ottiene:

$$\left[\gamma_0 - \gamma_h \frac{\beta_k}{\alpha_{kh}}\right] + \frac{\gamma_h}{\alpha_{kh}} x_{i_k} - \sum_{j=1,\ j\neq h}^{n-m} \left(\gamma_j - \gamma_h \frac{\alpha_{kj}}{\alpha_{kh}}\right) x_{i_{m+j}}.$$

Si ha ammissibilità per il duale se i coefficienti delle variabili x_{i_k} e $x_{i_{m+j}}$, $j = 1, \ldots, n-m,\ j \neq h$ sono non positivi. Per quanto riguarda il coefficiente di x_{i_k},

abbiamo

$$\underbrace{\frac{\overbrace{\gamma_h}^{\leq 0}}{\underbrace{\alpha_{kh}}_{>0}}}_{} \leq 0.$$

Per quanto riguarda i coefficienti di $x_{i_{m+j}}$, $j = 1, \ldots, n-m,\ j \neq h$ distinguiamo due casi: $\alpha_{kj} \leq 0$ e $\alpha_{kj} > 0$. Se $\alpha_{kj} \leq 0$, allora:

$$\gamma_j - \underbrace{\gamma_h}_{\leq 0} \underbrace{\frac{\overbrace{\alpha_{kj}}^{\leq 0}}{\alpha_{kh}}}_{>0} \leq \gamma_j \leq 0.$$

Se $\alpha_{kj} > 0$, allora:

$$\gamma_j - \gamma_h \frac{\alpha_{kj}}{\alpha_{kh}} = \underbrace{\alpha_{kj}}_{>0} \underbrace{\left(\frac{\gamma_j}{\alpha_{kj}} - \frac{\gamma_h}{\alpha_{kh}} \right)}_{\leq 0} \leq 0,$$

dove

$$\frac{\gamma_j}{\alpha_{kj}} - \frac{\gamma_h}{\alpha_{kh}} \leq 0$$

è una conseguenza della regola (5.13) per la scelta della variabile da far entrare in base.
Per quanto riguarda il nuovo valore dell'obiettivo, esso è pari a:

$$\gamma_0 - \underbrace{\gamma_h}_{\leq 0} \underbrace{\frac{\overbrace{\beta_k}^{< 0}}{\alpha_{kh}}}_{>0} \leq \gamma_0,$$

e quindi il valore dell'obiettivo per la nuova soluzione di base è migliore o quantomeno non peggiore rispetto alla precedente. Si noti che il duale è un problema di minimo e quindi per valore dell'obiettivo migliore si intende un valore inferiore rispetto al precedente. Si noti anche che se $\gamma_h < 0$ (cosa certamente vera nel caso non degenere) possiamo anche garantire che il nuovo valore dell'obiettivo sia strettamente minore rispetto al precedente.

Esempio 26 *Vediamo ora di risolvere un problema di PL utilizzando il simplesso duale. Si consideri il seguente problema di PL in forma standard:*

$$\begin{aligned} \max \quad & -3x_1 - 2x_2 + x_3 \\ & -2x_1 + x_2 + x_3 = -1 \\ & x_1 + 2x_2 + x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Prendiamo la riformulazione di tale problema rispetto alla base $\{x_3, x_4\}$. Si ottiene:

$$\begin{aligned} \max \quad & -1 - x_1 - 3x_2 \\ & x_3 = -1 + 2x_1 - x_2 \\ & x_4 = 2 - x_1 - 2x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \tag{5.14}$$

Si nota che la soluzione di base del primale associata alla base $\{x_3, x_4\}$ è:

$$x_1 = x_2 = 0 \quad x_3 = -1 \quad x_4 = 2,$$

e quindi non è ammissibile, mentre la soluzione di base del duale associata alla stessa base è ammissibile in quanto i coefficienti delle variabili fuori base x_1 e x_2 nell'obiettivo di (5.14) sono non positivi (rispettivamente pari a -1 e -3: essendo strettamente negativi possiamo anche dire che la soluzione di base del duale è non degenere). Quindi possiamo applicare il simplesso duale. Per prima cosa dobbiamo fare la verifica di ottimalità, che nel simplesso duale si riduce a controllare se i termini noti dei vincoli della riformulazione sono non negativi. Nel nostro caso ciò non è vero in quanto il termine noto del vincolo relativo alla variabile x_3 in (5.14) è pari a -1.
Successivamente dobbiamo verificare se l'obiettivo del problema duale è illimitato o, equivalentemente, se il problema primale ha regione ammissibile vuota. Perché questo accada si deve avere che in almeno una delle equazioni in cui il termine noto è negativo, i coefficienti delle variabili fuori base sono tutti non positivi. Ma in (5.14) l'unica equazione con termine noto negativo è quella relativa a x_3 e in questa equazione la variabile x_1 ha coefficiente positivo pari a 2. A questo punto dobbiamo scegliere la variabile da far uscire dalla base. La scelta ricade su quella con termine noto più piccolo e quindi sulla x_3.
Ora dobbiamo scegliere la variabile da far entrare in base. Utilizzando la regola (5.13), la scelta è ristretta alla sola variabile x_1, la sola ad avere coefficiente positivo nell'equazione relativa alla variabile x_3.
Infine dobbiamo eseguire l'operazione di cardine facendo uscire dalla base x_3 e facendovi entrare x_1. La riformulazione rispetto alla nuova base $\{x_1, x_4\}$ è la seguente:

$$\begin{aligned} \max \quad & -3/2 - 1/2x_3 - 7/2x_2 \\ & x_1 = 1/2 + 1/2x_3 + 1/2x_2 \\ & x_4 = 3/2 - 1/2x_3 - 3/2x_2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Come previsto, si mantiene l'ammissibilità duale. Ma ora si ha anche che la nuova base soddisfa la condizione di ottimalità (tutti i termini noti non negativi). Quindi la nuova base è ottima e il problema primale ha soluzione ottima:

$$x_1^* = 1/2 \quad x_4^* = 3/2 \quad x_2^* = x_3^* = 0,$$

con valore ottimo pari a $-3/2$.

Come esercizio si provi a risolvere il seguente problema di PL con il simplesso duale, partendo dalla base $\{x_3, x_4, x_5\}$*:*

$$\begin{aligned} \max \quad & -3x_1 - 2x_2 + x_3 \\ & 2x_1 + x_2 + x_3 = 2 \\ & x_1 + 2x_2 + x_4 = 2 \\ & -x_1 - x_2 + x_5 = -2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

5.2 Duale di un problema di PL generico

Abbiamo visto sino a ora come si determina il duale di un problema di PL in forma standard. è possibile però determinare il duale anche di problemi di PL in forma più generale. Si può fare ciò riconducendo dapprima il problema in forma standard, come già accennato in precedenza per l'individuazione del duale del problema duale (5.2). Tuttavia qui daremo alcune regole per determinare in modo diretto il duale di un problema di PL in forma generica, senza doversi ricondurre alla forma standard. Come per i problemi in forma standard, vi sarà una stretta relazione tra le variabili di un problema ed i vincoli dell'altro. Più precisamente avremo, come per la forma standard, che:

1. nel primale ci sono n variabili esattamente come nel duale vi sono n vincoli. Inoltre, i coefficienti del j-esimo vincolo del duale coincidono con i coefficienti della variabile x_j nei vincoli del primale, mentre il termine noto del j-esimo vincolo del duale coincide con il coefficiente di x_j nell'obiettivo del primale.

2. Nel primale vi sono m vincoli esattamente come nel duale vi sono m variabili. Inoltre, i coefficienti dell'i-esima variabile u_i del duale coincidono con i coefficienti dell'i-esimo vincolo del primale, mentre il coefficiente di u_i nell'obiettivo del duale coincide con il termine noto dell'i-esimo vincolo del primale.

Rispetto alla forma standard quello che può cambiare sono i versi delle disequazioni ed i segni delle variabili. Per stabilire questi ci si può rifare allo specchietto nella Tabella 5.1. Lo specchietto ci dice, per esempio, che se il primale è un problema di massimo ed una variabile è nel primale ≥ 0, allora il vincolo corrispondente del duale è di $\geq$, oppure che se il primale è un problema di minimo ed un vincolo del primale è di $=$, allora la variabile corrsipondente del duale è libera in segno (può assumere sia valori negativi che positivi). Come esercizio, si usi lo specchietto per verificare che il duale del seguente problema

Tabella 5.1:

min	max
variabile ≥ 0	vincolo $\leq$
variabile ≤ 0	vincolo $\geq$
variabile libera	vincolo $=$
vincolo $\geq$	variabile ≥ 0
vincolo $\leq$	variabile ≤ 0
vincolo $=$	variabile libera

di PL

$$\begin{array}{ll} \min & x_1 + 2x_2 + 3x_3 \\ & x_1 + x_2 - x_3 \leq 1 \\ & x_1 - 2x_2 + x_3 \geq 2 \\ & x_1 - x_2 - x_3 = 4 \\ & x_1 \geq 0,\ x_2 \leq 0,\ x_3 \text{ libera} \end{array} \tag{5.15}$$

è il seguente

$$\begin{array}{ll} \max & u_1 + 2u_2 + 4u_3 \\ & u_1 + u_2 + u_3 \leq 1 \\ & u_1 - 2u_2 - u_3 \geq 2 \\ & -u_1 + u_2 - u_3 = 3 \\ & u_1 \leq 0,\ u_2 \geq 0,\ u_3 \text{ libera} \end{array}$$

I risultati delle Osservazioni 15-18 e quelli del I e II teorema della dualità possono essere estesi anche a problemi di PL in forma più generale e ai loro duali. Ad esempio, a conferma della validità dell'Osservazione 18 si calcoli il duale del duale del problema (5.15) e si verifichi che coincide proprio con (5.15).

Capitolo 6

Analisi di sensitività

I coefficienti che appaiono in un problema di PL non sono sempre dei valori esatti ma piuttosto delle stime. Inoltre, anche quando si tratta di dati esatti, può accadere che dopo aver ottenuto la soluzione ottima del problema, un qualche dato del problema venga modificato. Questo ci spinge a porci la seguente domanda: se i coefficienti vengono modificati, come cambiano la soluzione ottima e il valore ottimo del nostro problema? In altre parole, la soluzione ottima e il valore ottimo del nostro problema quanto sono *sensibili* a modifiche dei valori dei coefficienti? L'*analisi di sensitività* si occupa della risposta a tali domande. Supponiamo di avere il nostro problema di PL in forma standard:

$$\begin{aligned} \max \quad & \mathbf{cx} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

e di aver stabilito che una base ottima del problema è B^*. Avremo quindi la seguente riformulazione rispetto alla base B^*:

$$\begin{aligned} \max \quad & \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} + (\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} - \mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned} \tag{6.1}$$

Avremo inoltre che una soluzione ottima del primale è:

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} \quad \mathbf{x}_{N^*} = 0,$$

mentre per il duale si ha la soluzione ottima:

$$\mathbf{u}^{B^*} = \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}.$$

Entrambi i problemi hanno valore ottimo:

$$\mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b}.$$

A questo punto vediamo cosa succede modificando diversi tipi di coefficienti. Ci limiteremo ad analizzare l'effetto di modifiche di singoli coefficienti ma l'analisi può essere estesa anche alla modifica in contemporanea di più coefficienti.

6.1 Modifica di un termine noto

Supponiamo che il termine noto b_r di un dato vincolo venga modificato in $b_r + \Delta b_r$, dove Δb_r è un certo valore reale (può essere sia positivo che negativo). Avremo quindi al posto del vettore $\mathbf{b}$ il vettore $\overline{\mathbf{b}}$ le cui componenti sono tutte uguali a quelle del vettore $\mathbf{b}$, tranne la r-esima che diventa $b_r + \Delta b_r$, ovvero

$$\overline{b}_i = b_i \quad \forall\, i \neq r, \quad \overline{b}_r = b_r + \Delta b_r.$$

Andando a sostituire nella riformulazione (6.1), si ottiene:

$$\begin{aligned} \max \quad & \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} + (\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} - \mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned}$$

Si possono avere due casi:

Caso 1 Il vettore $\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}}$ ha delle componenti negative. Ciò vuol dire che la modifica del termine noto ha reso la soluzione di base del primale associata a B^* non ammissibile. Tuttavia, la condizione

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

che non dipende dai termini noti, continua a essere soddisfatta e quindi la soluzione di base del duale associata a B^* continua a essere ammissibile. è importante sottolineare come *non sia necessario risolvere il nuovo problema da zero*, visto che, sfruttando l'ammissibilità della soluzione di base del duale associata a B^*, possiamo applicare il simplesso duale con base iniziale proprio B^*. In questo modo si arriva tipicamente a una soluzione del problema modificato in tempi molto più rapidi che non risolvendolo da zero.

Caso 2 Si ha che:

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} \geq 0.$$

Quindi la soluzione di base del primale associata a B^* continua ad essere ammissibile. Come nel Caso 1, la soluzione di base del duale associata a B^* resta ammissibile. Poiché entrambe le soluzioni di base sono ammissibili, si ha che la base B^* è ancora ottima. La nuova soluzione ottima del primale diventa

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} \quad \mathbf{x}_{N^*} = 0,$$

mentre quella del duale resta invariata e cioè pari a $\mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}$. Vediamo ora come cambia il valore ottimo rispetto a prima. Il nuovo valore ottimo è $\mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}}$ e quindi la differenza rispetto al precedente è pari a:

$$\mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} = \mathbf{u}^{B^*}(\overline{\mathbf{b}} - \mathbf{b}) = \sum_{i=1}^{m} u_i^{B^*}(\overline{b}_i - b_i).$$

Poiché:

$$\overline{b}_i = b_i \;\; \forall\, i \neq r \quad \overline{b}_r = b_r + \Delta b_r,$$

avremo:

$$\mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1} = u_r^{B^*}(\overline{b}_r - b_r) = u_r^{B^*}\Delta b_r.$$

Questo ci dà una interpretazione della soluzione ottima del duale: il valore della r-esima variabile nella soluzione ottima del duale misura la rapidità con cui cambia il valore ottimo del problema al variare del termine noto b_r del vincolo r-esimo del primale. Se il valore di $u_r^{B^*}$ è elevato, anche piccole modifiche di b_r possono portare a consistenti variazioni del valore ottimo. Se invece il valore di $u_r^{B^*}$ è piccolo, il valore ottimo del problema non è molto sensibile a variazioni del termine noto b_r.

Si noti che per il calcolo di $\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}}$ si può sfruttare il seguente risultato:

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} = \mathbf{A}_{B^*}^{-1}\mathbf{b} + \Delta b_r \mathbf{A}_{B^*}^{-1}\mathbf{e}_r$$

dove

$$\mathbf{e}_r = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

è il vettore che ha tutte componenti nulle tranne la r-esima che è pari a 1. Si noti che $\mathbf{A}_{B^*}^{-1}\mathbf{b}$ è noto, mentre $\mathbf{A}_{B^*}^{-1}\mathbf{e}_r$ equivale alla r-esima colonna della matrice $\mathbf{A}_{B^*}^{-1}$.

Possiamo anche porci la seguente domanda: in quale intervallo posso far variare Δb_r senza perdere l'ottimalità della base B^*? Basta imporre la seguente condizione

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} \geq 0$$

o, equivalentemente, in base a quanto appena visto, risolvere il sistema di disequazioni con incognita Δb_r

$$\Delta b_r \mathbf{A}_{B^*}^{-1}\mathbf{e}_r \geq -\mathbf{A}_{B^*}^{-1}\mathbf{b}.$$

6.2 Modifica del coefficiente di costo di una variabile al di fuori della base ottima

Supponiamo che venga modificato il coefficiente di costo di una delle variabili fuori base $x_{i_{m+j}}$, $j = 1, \ldots, n-m$, ad esempio la variabile $x_{i_{m+t}}$. Avremo quindi al posto del vettore $\mathbf{c}_{N^*}$ il vettore $\overline{\mathbf{c}}_{N^*}$ le cui componenti sono tutte uguali a

quelle del vettore $\mathbf{c}_{N^*}$, tranne quella relativa alla variabile $x_{i_{m+t}}$. Andando a sostituire nella riformulazione (6.1), si ottiene:

$$\begin{aligned} \max \quad & \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} + (\overline{\mathbf{c}}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} - \mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned}$$

Si possono avere ancora due casi:

Caso 1 Il vettore

$$\overline{\mathbf{c}}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*},$$

ha delle componenti positive. Questo vuol dire che la soluzione di base del duale associata a B^* non è più ammissibile. Resta invece ammissibile la soluzione di base del primale associata a B^*

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b}, \quad \mathbf{x}_{N^*} = 0,$$

che non è influenzata dalla modifica introdotta. Quindi, anche qui non è necessario risolvere il nuovo problema da zero, ma, sfruttando l'ammissibilità della soluzione di base del primale associata a B^*, possiamo applicare questa volta il simplesso primale con base iniziale proprio B^*. Si può anche dimostrare che la prima variabile che entrerà in base sarà proprio la variabile $x_{i_{m+t}}$. Perché?

Caso 2 Si ha che:

$$\overline{\mathbf{c}}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

Quindi la soluzione di base del duale associata a B^* continua ad essere ammissibile. Come nel Caso 1, la soluzione di base del primale associata a B^* resta ammissibile. Poiché entrambe le soluzioni di base sono ammissibili, si ha che la base B^* è ancora ottima. Le soluzioni ottime del primale e del duale restano invariate in quanto non dipendono dalla modifica introdotta. Lo stesso vale per il valore ottimo comune dei due problemi.

Anche qui possiamo anche porci la seguente domanda: in quale intervallo posso far variare $\Delta c_{i_{m+t}}$ senza perdere l'ottimalità della base B^*? Si tratterà di trovare le soluzioni di questo sistema di disequazioni (di fatto come vedremo si tratterà di risolvere una singola disequazione) con incognita $\Delta c_{i_{m+t}}$

$$\overline{\mathbf{c}}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

6.3 Modifica del coefficiente di costo di una variabile appartenente alla base ottima

Supponiamo che venga modificato il coefficiente di costo di una delle variabili in base x_{i_r}, $r = 1, \ldots, m$, ad esempio la variabile x_{i_s}, che passa da c_{i_s} a $c_{i_s} + \Delta c_{i_s}$

. Avremo quindi al posto del vettore $\mathbf{c}_{B^*}$ il vettore $\overline{\mathbf{c}}_{B^*}$ le cui componenti sono tutte uguali a quelle del vettore $\mathbf{c}_{B^*}$, tranne quella relativa alla variabile x_{i_s}. Andando a sostituire nella riformulazione (6.1), si ottiene:

$$\begin{aligned} \max \quad & \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} + (\mathbf{c}_{N^*} - \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} - \mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned}$$

Si possono avere ancora due casi:

Caso 1 Il vettore

$$\mathbf{c}_{N^*} - \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*},$$

ha delle componenti positive. Questo vuol dire che la soluzione di base del duale associata a B^* non è più ammissibile. Resta invece ammissibile la soluzione di base del primale associata a B^*

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b}, \quad \mathbf{x}_{N^*} = 0,$$

che non è influenzata dalla modifica introdotta. Quindi, al solito, non è necessario risolvere il nuovo problema da zero, ma, sfruttando l'ammissibilità della soluzione di base del primale associata a B^*, possiamo applicare il simplesso primale con base iniziale proprio B^*.

Caso 2 Si ha che:

$$\mathbf{c}_{N^*} - \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

Quindi la soluzione di base del duale associata a B^* continua ad essere ammissibile. Come nel Caso 1, la soluzione di base del primale associata a B^* resta ammissibile. Poiché entrambe le soluzioni di base sono ammissibili, si ha che la base B^* è ancora ottima. La soluzione ottima del primale, che non dipende dalla modifica introdotta, resta invariata, mentre la nuova soluzione ottima del duale è

$$\mathbf{u}^{B^*} = \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}.$$

Vediamo anche come cambia il valore ottimo comune dei due problemi.

$$\overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} = (\overline{\mathbf{c}}_{B^*} - \mathbf{c}_{B^*})\mathbf{A}_{B^*}^{-1}\mathbf{b} = (\overline{\mathbf{c}}_{B^*} - \mathbf{c}_{B^*})\mathbf{x}_{B^*} = \Delta c_{i_s} x_{i_s}^*.$$

Quindi la variazione è data dal prodotto tra la perturbazione Δc_{i_s} del coefficiente di costo della variabile x_{i_s} ed il valore della stessa variabile x_{i_s} in corrispondenza della soluzione ottima del primale.

Ancora possiamo anche porci la seguente domanda: in quale intervallo posso far variare Δc_{i_s} senza perdere l'ottimalità della base B^*? Si tratterà di trovare le soluzioni di questo sistema di disequazioni con incognita Δc_{i_s}

$$\mathbf{c}_{N^*} - \overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{A}_{N^*} \leq 0,$$

6.4 Modifica di un coefficiente della matrice $\mathbf{A}_{N^*}$

Supponiamo ora che venga modificato un coefficiente della matrice $\mathbf{A}_{N^*}$, la matrice ottenuta da $\mathbf{A}$ considerando le sole colonne relative a variabili fuori base. Sia $\overline{\mathbf{A}}_{N^*}$ la nuova matrice ottenuta dopo la modifica. Andando a sostituire nella riformulazione (6.1), si ottiene:

$$\begin{aligned} \max \quad & \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\mathbf{b} + (\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{A}}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b} - \mathbf{A}_{B^*}^{-1}\overline{\mathbf{A}}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned}$$

Si possono avere ancora due casi:

Caso 1 Il vettore

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{A}}_{N^*},$$

ha delle componenti positive. Questo vuol dire che la soluzione di base del duale associata a B^* non è più ammissibile. Resta invece ammissibile la soluzione di base del primale associata a B^*

$$\mathbf{x}_{B^*} = \mathbf{A}_{B^*}^{-1}\mathbf{b}, \quad \mathbf{x}_{N^*} = 0,$$

che non è influenzata dalla modifica introdotta. Quindi, anche qui non è necessario risolvere il nuovo problema da zero, ma, sfruttando l'ammissibilità della soluzione di base del primale associata a B^*, possiamo applicare il simplesso primale con base iniziale proprio B^*.

Caso 2 Si ha che:

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{A}}_{N^*} \leq 0,$$

Quindi la soluzione di base del duale associata a B^* continua ad essere ammissibile. Come nel Caso 1, la soluzione di base del primale associata a B^* resta ammissibile. Poiché entrambe le soluzioni di base sono ammissibili, si ha che la base B^* è ancora ottima. Le soluzioni ottime del primale e del duale restano invariate in quanto non dipendono dalla modifica introdotta. Lo stesso vale per il valore ottimo comune dei due problemi.

Come sempre possiamo porci la domanda: in quale intervallo posso far variare la perturbazione della componente di A_{N^*} senza perdere l'ottimalità della base B^*? Si tratterà di trovare le soluzioni di questo sistema di disequazioni (di fatto come vedremo si tratterà di risolvere una singola disequazione) con incognita la perturbazione della componente di A_{N^*}

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1}\overline{\mathbf{A}}_{N^*} \leq 0,$$

6.5 Modifica di un coefficiente della matrice $\mathbf{A}_{B^*}$

Supponiamo ora che venga modificato un coefficiente della matrice $\mathbf{A}_{B^*}$, la matrice ottenuta da $\mathbf{A}$ considerando le sole colonne relative a variabili nella base B^*. A questo caso accenniamo solo brevemente solo per far notare che è il più complicato tra quelli visti. Sia $\overline{\mathbf{A}}_{B^*}$ la nuova matrice ottenuta dopo la modifica. Il primo inconveniente che si può verificare è che $\overline{\mathbf{A}}_{B^*}$ non sia invertibile. A questo punto B^* non è più neppure una base. Per esempio, se abbiamo

$$\mathbf{A}_{B^*} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}$$

basta cambiare la componente $[A_{B^*}]_{11} = 2$ in $[A_{B^*}]_{11} = 0$ per rendere A_{B^*} non invertibile. Ma vediamo cosa può succedere anche nel caso fortunato in cui $\overline{\mathbf{A}}_{B^*}$ sia invertibile. Andando a sostituire nella riformulazione (6.1), si ottiene:

$$\begin{aligned} \max \quad & \mathbf{c}_{B^*}\overline{\mathbf{A}}_{B^*}^{-1}\mathbf{b} + (\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\overline{\mathbf{A}}_{B^*}^{-1}\mathbf{A}_{N^*})\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*} = \overline{\mathbf{A}}_{B^*}^{-1}\mathbf{b} - \overline{\mathbf{A}}_{B^*}^{-1}\mathbf{A}_{N^*}\mathbf{x}_{N^*} \\ & \mathbf{x}_{B^*}, \mathbf{x}_{N^*} \geq 0 \end{aligned}$$

Si può avere che il vettore:

$$\overline{\mathbf{A}}_{B^*}^{-1}\mathbf{b}$$

ha componenti negative, mentre il vettore

$$\mathbf{c}_{N^*} - \mathbf{c}_{B^*}\overline{\mathbf{A}}_{B^*}^{-1}\mathbf{A}_{N^*}$$

ha componenti positive. In altre parole, né la soluzione di base del primale associata a B^*, né quella del duale sono ammissibili. Di conseguenza, non possiamo applicare né il simplesso duale né quello primale con base iniziale B^*. Esistono comunque modi per non dover tornare a risolvere da zero il problema modificato ma qui non li esploreremo.

6.6 L'esempio del problema degli aiuti umanitari

Nel Capitolo 1 abbiamo introdotto il problema degli aiuti umanitari, che ha la seguente formulazione come problema di PL:

$$\begin{aligned} \max \quad & 14x_1 + 5x_2 + 4x_3 \\ & 10x_1 + 30x_2 + 20x_3 \leq 5100 \\ & 10x_1 + 20x_2 + 40x_3 \leq 8000 \\ & 30x_1 + 10x_2 + 5x_3 \leq 1805 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Il problema è in forma canonica ma con l'aggiunta delle variabili non negative y_1, y_2, y_3 lo si può ricondurre alla forma standard:

$$\begin{aligned} \max \quad & 14x_1 + 5x_2 + 4x_3 \\ & 10x_1 + 30x_2 + 20x_3 + y_1 = 5100 \\ & 10x_1 + 20x_2 + 40x_3 + y_2 = 8000 \\ & 30x_1 + 10x_2 + 5x_3 + y_3 = 1805 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned} \tag{6.2}$$

Partendo dalla base ammissibile $B_0 = \{y_1, y_2, y_3\}$ si può applicare il metodo del simplesso e si giunge alla base ottima $B^* = \{y_1, x_3, x_1\}$ con la seguente riformulazione rispetto ad essa:

$$\begin{aligned} \max \quad & 1164 - \tfrac{1}{23}y_2 - \tfrac{52}{115}y_3 - \tfrac{9}{23}x_2 \\ & y_1 = 960 + \tfrac{4}{23}y_3 + \tfrac{11}{23}y_2 - \tfrac{430}{23}x_2 \\ & x_3 = 193 + \tfrac{1}{115}y_3 - \tfrac{3}{115}y_2 - \tfrac{10}{23}x_2 \\ & x_1 = 28 - \tfrac{4}{115}y_3 + \tfrac{1}{230}y_2 - \tfrac{6}{23}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned} \tag{6.3}$$

Da qui si vede immediatamente che la soluzione ottima (unica) del primale è:

$$y_1^* = 960 \quad x_3^* = 193 \quad x_1^* = 28 \quad x_2^* = y_2^* = y_3^* = 0.$$

Essendo partiti con la base B_0 le cui variabili formano la matrice identica nella formulazione originaria (6.2), posso, con il metodo visto in precedenza, leggere la matrice $\mathbf{A}_{B^*}^{-1}$ dalla riformulazione (6.3):

$$\mathbf{A}_{B^*}^{-1} = \begin{bmatrix} 1 & -\frac{11}{23} & -\frac{4}{23} \\ 0 & \frac{3}{115} & -\frac{1}{115} \\ 0 & -\frac{1}{230} & \frac{4}{115} \end{bmatrix}$$

La soluzione di base del duale $\mathbf{u}^{B^*}$ associata a B^* sarà soluzione ottima del duale:

$$\mathbf{u}^{B^*} = \mathbf{c}_{B^*}\mathbf{A}_{B^*}^{-1} = (0 \;\; 4 \;\; 14)\mathbf{A}_{B^*}^{-1} = \begin{pmatrix} 0 & \frac{1}{23} & \frac{52}{115} \end{pmatrix}$$

(per esercizio si ricavi la stessa soluzione anche attraverso le condizioni di complementarità).

6.6.1 Modifiche nei dati del problema

Tra i dati del problema degli aiuti umanitari alcuni sono esatti mentre altri sono delle stime. In particolare, la disponibilità di farina, acqua e medicinali e i contenuti di ciascun tipo di pacco sono dati esatti, mentre i valori di utilità sono stime. Ora, se ho a che fare con dati stimati è importante chiedersi quanto la mia soluzione sia sensibile al valore di tali dati. Per esempio, il valore stimato dell'utilità del pacco di tipo I è 14 ma mi posso chiedere come cambiano la

soluzione ottima e il valore ottimo del problema se il valore di utilità passa da 14 a 13 oppure a 15. Se queste modifiche lasciano sostanzialmente invariata la soluzione ottima del mio problema, allora vuol dire che questa è poco sensibile a modifiche del dato stimato e di conseguenza possiamo aspettarci conclusioni corrette anche usando una stima non troppo precisa. Viceversa, se le modifiche comportano notevoli cambiamenti nella soluzione ottima, allora vuol dire che questa è molto sensibile a modifiche del dato ed in tal caso per avere conclusioni corrette occorre avere una stima molto precisa del dato. Quindi, in presenza di dati stimati è importante riuscire a capire che cosa succede quando apporto a essi delle modifiche.

Ma non è questo l'unico caso in cui voglio sapere cosa succede se modifico un qualche dato. Nel nostro esempio degli aiuti umanitari posso, dopo aver risolto il problema, chiedermi cosa succede se riesco a procurarmi, ad esempio, altri 100 sacchi di farina portando quindi la disponibilità di questa da 5100 a 5200 sacchi. Anche in questo caso vogliamo vedere quale impatto avrebbe sulla soluzione ottima del nostro problema la modifica di un dato del problema stesso.

L'analisi di sensitività si occupa proprio di studiare gli effetti di modifiche nei dati del problema e vedremo ora come funziona modificando singolarmente alcuni dati del nostro problema.

Modifica di un termine noto

Vediamo cosa succede se modifico la disponibilità di acqua (che attualmente è $b_2 = 8000$) di Δb_2. In generale avremo che

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} = (960 \quad 193 \quad 28) \; + \; \Delta b_2 \left(-\frac{11}{23} \quad \frac{3}{115} \quad -\frac{1}{230} \right) =$$

In particolare, vediamo cosa succede prima incrementandola di 1150 unità ($\Delta b_2 = 1150$) e poi di 2300 unità ($\Delta b_2 = 2300$).

$\Delta b_2 = 1150$ Il nuovo vettore di termini noti sarà:

$$\overline{\mathbf{b}} = (5100 \quad 9150 \quad 1805).$$

Nella riformulazione rispetto a B^* le uniche parti che vengono modificate sono i termini noti dei vincoli che diventeranno:

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} = (410 \quad 223 \quad 23)$$

e il valore dell'obiettivo, che viene modificato della quantità:

$$u_2^* \Delta b_2 = \frac{1}{23} 1150 = 50,$$

e quindi passa da 1164 a 1214. La soluzione di base del primale associata a B^* è ancora ammissibile. Resta invariata la soluzione di base del duale $\mathbf{u}^{B^*}$

associata a B^* e quindi continua ad essere ammissibile. Quindi le due soluzioni sono ottime per i rispettivi problemi. La nuova soluzione ottima del primale è:

$$x_1^* = 223 \quad x_3^* = 23 \quad y_1^* = 410 \quad x_2^* = y_2^* = y_3^* = 0.$$

Il valore ottimo dei due problemi è ora 1214.

$\Delta b_2 = 2300$ Il nuovo vettore di termini noti sarà:

$$\overline{\mathbf{b}} = (5100 \quad 10300 \quad 1805).$$

Nella riformulazione rispetto a B^* le uniche parti che vengono modificate sono i termini noti dei vincoli che diventeranno:

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} = (-140 \quad 253 \quad 18)$$

e il valore dell'obiettivo, che viene modificato della quantità:

$$u_2^* \Delta b_2 = \frac{1}{23} 2300 = 100,$$

e quindi passa da 1164 a 1264. La riformulazione rispetto a tale base sarà la seguente:

$$\begin{array}{ll} \max & 1264 - \frac{1}{23}y_2 - \frac{52}{115}y_3 - \frac{9}{23}x_2 \\ & y_1 = -140 + \frac{4}{23}y_3 + \frac{11}{23}y_2 - \frac{430}{23}x_2 \\ & x_3 = 253 + \frac{1}{115}y_3 - \frac{3}{115}y_2 - \frac{10}{23}x_2 \\ & x_1 = 18 - \frac{4}{115}y_3 + \frac{1}{230}y_2 - \frac{6}{23}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{array}$$

La soluzione di base del primale associata a B^* non è più ammissibile. Ma la soluzione di base del duale $\mathbf{u}^{B^*}$ associata a B^* resta invariata e continua ad essere ammissibile. Quindi possiamo applicare il simplesso duale con base iniziale B^*.

Per determinare l' intervallo dove possiamo far variare Δb_2 senza che cambi la base ottima, dobbiamo risolvere il seguente sistema di disequazioni

$$\begin{cases} 960 - \frac{11\Delta b_2}{23} \geq 0 \\ 193 + \frac{3\Delta b_2}{115} \geq 0 \\ 28 - \frac{\Delta b_2}{230} \geq 0 \end{cases}$$

Vediamo anche che cosa succede se apporto la modifica $\Delta b_1 = 100$, ovvero incremento di 100 unità la disponibilità di sacchi di farina. Si può verificare che il nuovo vettore di termini noti sarà:

$$\overline{\mathbf{b}} = (5200 \quad 8000 \quad 1805).$$

Nella riformulazione rispetto a B^* le uniche parti che vengono modificate sono i termini noti dei vincoli che diventeranno:

$$\mathbf{A}_{B^*}^{-1}\overline{\mathbf{b}} = (1060 \quad 193 \quad 28)$$

Quindi la base B^* è ancora ottima. Il valore dell'obiettivo viene modificato della quantità:

$$u_1^*\Delta b_1 = 0 * 100 = 0,$$

e quindi resta invariato. Ciò ci dice che avere 100 sacchi di farina in più non migliora le cose. Questo lo si poteva intuire anche analizzando quello che succedeva in corrispondenza della soluzione ottima trovata. Mentre in tale soluzione acqua e medicinali venivano usati in modo completo, restavano invece inutilizzati 960 sacchi di farina. Ne consegue che, se già con 5100 sacchi di farina si avevano delle eccedenze, non è aumentando il numero di tali sacchi che possiamo sperare di migliorare il valore ottimo, ma piuttsosto dovremo incrementare il numero di bottiglie d'acqua e/o di scatole di medicinali.

Modifica del coefficiente di costo di una variabile fuori base

Consideriamo ora una modifica Δc_{x_2} del coefficiente di x_2 nell'obiettivo (il valore di utilità dei pacchi di tipo II). L'unica modifica nella riformulazione rispetto a B^* si avrà nell'obiettivo che diventerà

$$1164 - \frac{1}{23}y_2 - \frac{52}{115}y_3 - \frac{9}{23}x_2 + \Delta c_{x_2}x_2.$$

Vediamo cosa succede per $\Delta c_{x_2} = -1$ e $\Delta c_{x_2} = 1$.

$\Delta c_{x_2} = -1$ La nuova riformulazione rispetto a B^* è la seguente:

$$\begin{aligned} \max \quad & 1164 - \tfrac{1}{23}y_2 - \tfrac{52}{115}y_3 - \left(\tfrac{9}{23} + 1\right) x_2 \\ & y_1 = 960 + \tfrac{4}{23}y_3 + \tfrac{11}{23}y_2 - \tfrac{430}{23}x_2 \\ & x_3 = 193 + \tfrac{1}{115}y_3 - \tfrac{3}{115}y_2 - \tfrac{10}{23}x_2 \\ & x_1 = 28 - \tfrac{4}{115}y_3 + \tfrac{1}{230}y_2 - \tfrac{6}{23}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned}$$

La soluzione di base del duale associata a B^* è ancora ammissibile. Resta invariata la soluzione di base del primale associata a B^* e quindi continua ad essere ammissibile. Quindi le due soluzioni sono ottime per i rispettivi problemi. Le due soluzioni ottime ed il valore ottimo non subiscono modifiche.

$\Delta c_{x_2} = 1$ La nuova riformulazione rispetto a B^* è la seguente:

$$\begin{aligned} \max \quad & 1164 - \tfrac{1}{23}y_2 - \tfrac{52}{115}y_3 + \left(1 - \tfrac{9}{23}\right) x_2 \\ & y_1 = 960 + \tfrac{4}{23}y_3 + \tfrac{11}{23}y_2 - \tfrac{430}{23}x_2 \\ & x_3 = 193 + \tfrac{1}{115}y_3 - \tfrac{3}{115}y_2 - \tfrac{10}{23}x_2 \\ & x_1 = 28 - \tfrac{4}{115}y_3 + \tfrac{1}{230}y_2 - \tfrac{6}{23}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned}$$

La soluzione di base del duale associata a B^* non è più ammissibile. Ma la soluzione di base del primale associata a B^* resta invariata e continua ad essere ammissibile. Quindi possiamo applicare il simplesso primale con base iniziale B^*. Possiamo anche dire che la prima variabile ad entrare in base sarà certamente la x_2 (è l'unica con coefficiente di costo ridotto positivo).

Per determinare l'intervallo in cui può variare Δc_{x_2} senza cambiare la base ottima basta risolvere la singola disequazione:

$$-\frac{9}{23} + \Delta c_{x_2} \leq 0$$

Modifica del coefficiente di costo di una variabile appartenente alla base

Consideriamo ora una modifica Δc_{x_1} del coefficiente di x_1 nell'obiettivo (il valore di utilità dei pacchi di tipo I). L'unica modifica nella riformulazione rispetto a B^* si avrà nell'obiettivo che diventerà

$$1164 - \frac{1}{23}y_2 - \frac{52}{115}y_3 - \frac{9}{23}x_2 + \Delta c_{x_1} x_1.$$

Rispetto al caso precedente, qui però dobbiamo anche sostituire la variabile in base x_1 con la relativa espressione data dall'equazione

$$x_1 = 28 - \frac{4}{115}y_3 + \frac{1}{230}y_2 - \frac{6}{23}x_2.$$

Con tale sostituzione si arriva all'obiettivo

$$1164+28\Delta c_{x_1}+y_2\left(-\frac{1}{23}+\frac{\Delta c_{x_1}}{230}\right)+y_3\left(-\frac{52}{115}-\frac{4\Delta c_{x_1}}{115}\right)+x_2\left(-\frac{9}{23}-\frac{6\Delta c_{x_1}}{23}\right)$$

Vediamo cosa succede per $\Delta c_{x_1} = 1$ e $\Delta c_{x_1} = -2$.

$\Delta c_{x_1} = 1$ La nuova riformulazione rispetto a B^* è la seguente:

$$\begin{aligned} \max \quad & 1192 - \tfrac{9}{230}y_2 - \tfrac{56}{115}y_3 - \tfrac{15}{23}x_2 \\ & y_1 = 960 + \tfrac{4}{23}y_3 + \tfrac{11}{23}y_2 - \tfrac{430}{23}x_2 \\ & x_3 = 193 + \tfrac{1}{115}y_3 - \tfrac{3}{115}y_2 - \tfrac{10}{23}x_2 \\ & x_1 = 28 - \tfrac{4}{115}y_3 + \tfrac{1}{230}y_2 - \tfrac{6}{23}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned}$$

La soluzione di base del duale associata a B^* è ancora ammissibile. Resta invariata la soluzione di base del primale associata a B^* e quindi continua ad essere ottima. Il nuovo valore dell'obiettivo varia rispetto al precedente della quantità $\Delta c_{x_1} x_1^* = 1 * 28 = 28$. La nuova soluzione ottima del duale sarà data da

$$\overline{\mathbf{c}}_{B^*}\mathbf{A}_{B^*}^{-1} = \begin{pmatrix} 0 & \frac{9}{230} & \frac{56}{115} \end{pmatrix}$$

$\Delta c_{x_1} = -2$ La nuova riformulazione rispetto a B^* è la seguente:

$$\begin{aligned} \max \quad & 1108 - \tfrac{12}{230} y_2 - \tfrac{44}{115} y_3 + \tfrac{3}{23} x_2 \\ & y_1 = 960 + \tfrac{4}{23} y_3 + \tfrac{11}{23} y_2 - \tfrac{430}{23} x_2 \\ & x_3 = 193 + \tfrac{1}{115} y_3 - \tfrac{3}{115} y_2 - \tfrac{10}{23} x_2 \\ & x_1 = 28 - \tfrac{4}{115} y_3 + \tfrac{1}{230} y_2 - \tfrac{6}{23} x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{aligned}$$

In tal caso B^* non è più ammissibile per il duale. Ma la soluzione di base del primale associata a B^* resta invariata e continua ad essere ammissibile. Quindi possiamo applicare il simplesso primale con base iniziale B^*.

L'intervallo in cui può variare Δc_{x_1} senza cambiare la base ottima sarà dato dalla soluzione di questo sistema:

$$\begin{cases} -\frac{1}{23} + \frac{\Delta c_{x_1}}{230} \leq 0 \\ -\frac{52}{115} - \frac{4\Delta c_{x_1}}{115} \leq 0 \\ -\frac{9}{23} - \frac{6\Delta c_{x_1}}{23} \leq 0 \end{cases}$$

Modifica di un coefficiente della matrice $\mathbf{A}_{N^*}$

Vediamo cosa succede se modifico il numero di bottiglie d'acqua nei pacchi di tipo II (attualmente pari a 20) delle quantità $\Delta a_{22} = -10$ e $\Delta a_{22} = 10$. Quindi modifico il coefficiente della variabile x_2 nel secondo vincolo (quello dell'acqua) delle quantità indicate. Nella nuova riformulazione rispetto a B^* l'espressione dell'obiettivo viene modificato come segue: si prende il valore ottimo della variabile duale associata al vincolo in cui si è apportata la modifica (qui $u_2^{B^*} = \frac{1}{23}$); si moltiplica tale valore per la modifica apportata (Δa_{22}) e per la variabile il cui coefficiente è stato modificato (qui x_2); si sottrae il risultato $u_2^{B^*} \Delta a_{22} x_2$ nell'espressione dell'obiettivo. Nella riformulazione dovrò anche modificare i coefficienti di x_2 nei vincoli (tutti gli altri coefficienti restano invariati). I nuovi coefficienti sono dati dal seguente prodotto:

$$-\mathbf{A}_{B^*}^{-1} \begin{pmatrix} 30 \\ 20 + \Delta a_{22} \\ 10 \end{pmatrix}$$

$\Delta a_{22} = 10$ Si ha

$$-u_2^{B^*} \Delta a_{22} x_2 = -\frac{10}{23} x_2$$

e

$$-\mathbf{A}_{B^*}^{-1} \begin{pmatrix} 30 \\ 30 \\ 10 \end{pmatrix} = \begin{pmatrix} -\frac{320}{23} \\ -\frac{80}{115} \\ -\frac{25}{115} \end{pmatrix}$$

Quindi la riformulazione rispetto a B^* è la seguente:

$$\begin{array}{rc} \max & 1164 - \frac{1}{23}y_2 - \frac{52}{115}y_3 - \frac{19}{23}x_2 \\ & y_1 = 960 + \frac{4}{23}y_3 + \frac{11}{23}y_2 - \frac{320}{23}x_2 \\ & x_3 = 193 + \frac{1}{115}y_3 - \frac{3}{115}y_2 - \frac{80}{115}x_2 \\ & x_1 = 28 - \frac{4}{115}y_3 + \frac{1}{230}y_2 - \frac{25}{115}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{array}$$

La base B^* è ancora ottima. Le soluzioni ottime del primale e del duale rimangono invariate e lo stesso accade al valore ottimo.

$\Delta a_{22} = -10$ Si ha

$$-u_2^{B^*} \Delta a_{22} x_2 = \frac{10}{23} x_2$$

e

$$-A_{B^*}^{-1} \begin{pmatrix} 30 \\ 10 \\ 10 \end{pmatrix} = \begin{pmatrix} -\frac{540}{23} \\ -\frac{20}{115} \\ -\frac{35}{115} \end{pmatrix}$$

Quindi la riformulazione rispetto a B^* è la seguente:

$$\begin{array}{rc} \max & 1164 - \frac{1}{23}y_2 - \frac{52}{115}y_3 + \frac{1}{23}x_2 \\ & y_1 = 960 + \frac{4}{23}y_3 + \frac{11}{23}y_2 - \frac{540}{23}x_2 \\ & x_3 = 193 + \frac{1}{115}y_3 - \frac{3}{115}y_2 - \frac{20}{115}x_2 \\ & x_1 = 28 - \frac{4}{115}y_3 + \frac{1}{230}y_2 - \frac{35}{115}x_2 \\ & x_1, x_2, x_3, y_1, y_2, y_3 \geq 0 \end{array}$$

La base B^* non è più ottima ma è ancora ammissibile per il primale. Quindi posso applicare il simplesso primale a partire da tale base.

L'intervallo in cui può variare Δa_{22} senza cambiare la base ottima è dato dalla soluzione di questa disequazione:

$$-\frac{9}{23} - u_2^* \Delta a_{22} = -\frac{9}{23} - \frac{\Delta a_{22}}{23} \leq 0$$

Capitolo 7

Programmazione Lineare Intera

Fino a ora abbiamo affrontato problemi lineari in cui le variabili potevano assumere valori reali. Ora invece ci concentreremo su problemi in cui le variabili possono assumere solo valori interi. Questi problemi vengono chiamati problemi di Programmazione Lineare Intera (abbreviata con PLI nel seguito). Nel seguito ci occuperemo di alcuni risultati teorici sulla PLI e introdurremo due approcci di risoluzione per questo tipo di problemi.

7.1 Relazione tra un problema lineare intero e il suo rilassamento lineare

Si consideri un problema di PLI in forma standard:

$$\begin{aligned} w^* = \max \quad & \mathbf{cx} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0, \quad \mathbf{x} \in Z^n \end{aligned}$$

dove Z denota l'insieme degli interi. Si indicherà con Z_a la regione ammissibile di questo problema, e quindi

$$Z_a = \{\mathbf{x} \in Z^n \ : \ \mathbf{Ax} = \mathbf{b}, \ \mathbf{x} \geq 0\},$$

e con Z_{ott} l'insieme delle sue soluzioni ottime:

$$Z_{ott} = \{\mathbf{x}^* \in Z_a \ : \ \mathbf{cx}^* \geq \mathbf{cx} \ \ \forall \ \mathbf{x} \in Z_a\}.$$

Chiameremo *rilassamento lineare* del problema di PLI, il problema di PL ottenuto dal problema di PLI omettendo la richiesta che le variabili siano intere,

e quindi

$$\begin{aligned} z^* = \max \quad & \mathbf{cx} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

Esempio 27 *Si consideri il seguente problema di PLI:*

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 + 2x_2 \leq 4 \\ & 2x_1 + x_2 \leq 4 \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{aligned}$$

Si determinino per via grafica la soluzione del problema di PLI e del suo rilassamento lineare.

Come al solito, indicheremo con S_a e S_{ott} rispettivamente la regione ammissibile e l'insieme delle soluzioni ottime del rilassamento lineare del problema di PLI. Possiamo notare che

$$Z_a \subseteq S_a$$

e che i due problemi hanno la stessa funzione obiettivo $\mathbf{cx}$. Da ciò conseguono le seguenti relazioni tra il problema di PLI e il problema di PL suo rilassamento lineare.

1. Se $S_a = \emptyset$, allora $Z_a = \emptyset$.
2. Se l'obiettivo del problema di PLI è illimitato, allora lo è anche quello del suo rilassamento lineare.
3. Se $S_{ott} \neq \emptyset$ e $Z_{ott} \neq \emptyset$, allora il valore ottimo w^* del problema di PLI non può essere superiore al valore ottimo z^* del suo rilassamento lineare, ovvero
$$w^* \leq z^*$$
Infatti:
$$\begin{aligned} \bar{\mathbf{x}} \in Z_{ott} \quad \Rightarrow \quad \bar{\mathbf{x}} \in Z_a \quad \Rightarrow \quad \bar{\mathbf{x}} \in S_a \quad \Rightarrow \\ \Rightarrow \quad \mathbf{c}\bar{\mathbf{x}} \leq \mathbf{cx}^* \quad \text{per } \mathbf{x}^* \in S_{ott}. \end{aligned}$$
4. Se $S_{ott} \neq \emptyset$ contiene un punto x^* a coordinate tutte intere, allora $\mathbf{x}^* \in Z_{ott}$ e $w^* = z^*$. Infatti
$$\begin{aligned} \mathbf{x}^* \in S_{ott} \quad \Rightarrow \quad \mathbf{cx}^* \geq \mathbf{cx} \ \ \forall\, \mathbf{x} \in S_a \quad \Rightarrow \quad \mathbf{cx}^* \geq \mathbf{cx} \ \ \forall\, \mathbf{x} \in Z_a \\ \mathbf{x}^* \ \text{ a coordinate intere } \quad \Rightarrow \quad \mathbf{x}^* \in Z_a \end{aligned}$$
e quindi $\mathbf{x}^* \in Z_{ott}$. Una situazione del genere si ha nel seguente esempio
$$\begin{aligned} \max \quad & x_2 \\ & x_1 + 2x_2 \leq 4 \\ & 2x_1 + x_2 \leq 4 \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{aligned}$$

Altri casi possibili nelle relazioni tra un problema di PLI e il suo rilassamento lineare sono i seguenti.

1. $Z_a = \emptyset$ ma $S_{ott} \neq \emptyset$, come nel seguente esempio:

$$\begin{array}{ll} \max & x_2 \\ & x_1 \geq \frac{1}{4} \\ & x_1 \leq \frac{3}{4} \\ & x_2 \leq 2 \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{array}$$

2. $Z_a = \emptyset$ ma l'obiettivo del rilassamento lineare è illimitato, come nel seguente esempio:

$$\begin{array}{ll} \max & x_2 \\ & x_1 \geq \frac{1}{4} \\ & x_1 \leq \frac{3}{4} \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{array}$$

3. Se $\mathbf{A}, \mathbf{b}$ e $\mathbf{c}$ contengono solo valori razionali, allora $Z_{ott} \neq \emptyset$ implica $S_{ott} \neq \emptyset$. Se vi sono coefficienti irrazionali allora può accadere che $Z_{ott} \neq \emptyset$ ma il rilassamento lineare ha obiettivo illimitato, come nel seguente esempio:

$$\begin{array}{ll} \max & x_2 \\ & x_2 = \sqrt{2} x_1 \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{array}$$

   dove Z_a (e quindi Z_{ott}) contiene la sola origine, ma l'obiettivo del rilassamento lineare è illimitato.

Un'importante osservazione è la seguente.

I problemi di PL sono in generale molto più semplici e rapidi da risolvere dei problemi di PLI [1]*. In particolare il rilassamento lineare di un problema di PLI è tipicamente molto più facile da risolvere del problema di PLI stesso.*

Questa osservazione può spingere a risolvere i problemi di PLI con la seguente procedura:

- risolvo il rilassamento lineare del problema di PLI;
- arrotondo le variabili nella soluzione ottima che non hanno valore intero ma decimale (se ne esistono: nel caso in cui non ne esistano la soluzione del rilassamento lineare è anche soluzione del problema di PLI).

[1] Nella teoria della complessità, che tratteremo nel Capitolo 10, i problemi di PL sono classificati tra quelli risolvibili in tempo polinomiale, mentre quelli di PLI sono verosimilmente solo risolvibili in tempo esponenziale

In realtà questa procedura rischia di restituire risultati molto inesatti. Ad esempio, in precedenza abbiamo visto un caso dove il rilassamento lineare ha soluzione ottima mentre il problema di PLI ha regione ammissibile vuota. In tal caso, applicando la procedura vista sopra, restituiremmo una soluzione ottima per un problema che non ha neppure una soluzione ammissibile. Va detto comunque che una procedura di questo tipo è accettabile quando si ha a che fare con variabili che assumono valori molto elevati. Se nella soluzione ottima del rilassamento lineare ho una variabile con valore pari a 20000.4 e la arrotondo a 20000, posso introdurre un errore ma essendo la parte decimale (0.4) quasi irrilevante rispetto al valore 20000.4 della variabile, tale errore è tipicamente trascurabile. Al contrario, con variabili intere che assumono valori piccoli, ad esempio 1.4, l'arrotondamento a 1 del valore di tale variabile può causare errori non trascurabili, come conseguenza del fatto che la parte decimale (0.4) non è affatto irrilevante rispetto al valore di 1.4. Questo è in particolare vero quando si ha a che fare con variabili binarie, con le quali la procedura vista sopra va certamente evitata.

Restano allora da indicare procedure di risoluzione per i problemi di PLI applicabili in tutti i casi. Le vedremo nei capitoli successivi. Prima di esse, affrontiamo un ulteriore argomento teorico che ci permetterà di individuare sottoclassi di problemi di PLI più semplici rispetto ai generici problemi di PLI.

7.2 Chiusure convesse e sottoclassi speciali della PLI

Come già accennato in precedenza e come vedremo più in dettaglio nel Capitolo 10, i problemi di PLI sono molto più difficili da risolvere rispetto ai problemi di PL. In particolare il rilassamento lineare di un problema di PLI è generalmente risolvibile in tempi molto più rapidi rispetto al problema di PLI stesso. Questo fatto viene sfruttato algoritmicamente, visto che sia gli algoritmi di taglio che quelli branch-and-bound per la risoluzione di problemi di PLI si basano sulla risoluzione multipla di problemi di PLI (come vedremo nei Capitoli 8 e 9).
In questo capitolo partendo dalla definizione di un problema di PL equivalente a un problema di PLI, basata sul concetto di chiusura convessa, arriveremo a identificare sottoclassi 'facili' dei problemi di PLI. In particolare, si tratterà di problemi di PLI risolvibili come problemi di PL semplicemente rimuovendo i vincoli di interezza sulla variabili.

7.2.1 Problemi di PLI e chiusure convesse

Supponiamo di avere il nostro problema di PLI

$$\max_{x \in Z_a} cx. \tag{7.1}$$

La relazione tra regione ammissibile Z_a del problema di PLI e regione ammissibile S_a del suo rilassamento lineare è la seguente

$$Z_a = S_a \cap Z^n$$

Diamo ora la definizione di *chiusura convessa* di un insieme.

Definizione 1 *Dato un insieme T, la chiusura convessa di T, indicata con $conv(T)$, è il più piccolo insieme convesso (si veda la Definzione 1) contenente T.*

In Figura 7.1 è riportato un insieme T e la sua chiusura convessa. Si consideri

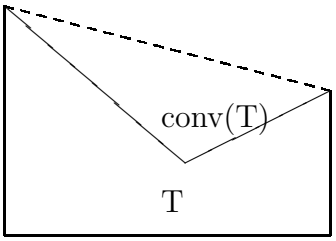


Figura 7.1: Un insieme T e la sua chiusura convessa.

ora la chiusura convessa $conv(Z_a)$ di Z_a. Si può dimostrare che, se $Z_a \neq \emptyset$, $conv(Z_a)$ è un poliedro, cioè esiste una matrice A' ed un vettore b' tale che

$$conv(Z_a) = \{x \in R^n : \ A'x \leq b', \ x \geq 0\}. \tag{7.2}$$

Si consideri ora il seguente problema di PL

$$\max_{x \in conv(Z_a)} cx \tag{7.3}$$

Si può dimostrare che:

- il problema (7.1) ammette soluzione se e solo se la ammette il problema (7.3);
- ogni soluzione ottima del problema (7.1) è soluzione ottima del problema (7.3);
- esiste almeno una soluzione ottima (di base) del problema (7.3) che è anche soluzione ottima del problema (7.1).

Questo vuol dire che se conoscessimo la matrice A' ed il vettore b' che definiscono $conv(Z_a)$, potremmo risolvere il nostro problema di PLI risolvendo il problema di PL (7.3). Il problema è che in molti casi $conv(Z_a)$ non è noto e determinarlo è difficile almeno quanto risolvere il problema di PLI stesso. Citiamo anche il fatto che in alcuni casi $conv(Z_a)$ è noto ma è formato da un numero esponenziale di vincoli, il che può rendere inefficiente la risoluzione del problema (7.3).

Esempio 28 *Si consideri il seguente problema di PLI,*

$$\begin{array}{l} \max x_1 + x_2 \\ x_1 + \dfrac{1}{2}x_2 \leq \dfrac{7}{4} \\ \dfrac{1}{2}x_1 + x_2 \leq \dfrac{7}{4} \\ x_1, x_2 \geq 0 \text{ interi} \end{array}$$

Si ha $S_a = \{(x_1, x_2) : \ Ax \leq b, \ x_1, x_2 \geq 0\}$, *con*

$$A = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \qquad b = \begin{bmatrix} \frac{7}{4} \\ \frac{7}{4} \end{bmatrix}.$$

L'insieme $Z_a = S_a \cap Z^2$ *è formato dai quattro punti*

$$(0,0) \quad (0,1) \quad (1,0) \quad (1,1)$$

Si ha che

$$conv(Z_a) = \{(x_1, x_2) : \ x_1 \leq 1, \ x_2 \leq 1, \ x_1, x_2 \geq 0\}.$$

(vedi Figura 7.2). In tal caso

$$A' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad b' = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Si noti che indipendentemente dalla funzione obiettivo i problemi (7.1) e (7.3) hanno sempre soluzione in questo caso. Con la funzione obiettivo $x_1 + x_2$ *entrambi i problemi hanno l'unica soluzione* $(1,1)$. *Se si considera invece l'obiettivo* x_1, *si ha che il problema (7.1) ha soluzioni* $(1,0)$ *e* $(1,1)$, *mentre il problema (7.3) ha come soluzioni l'intero segmento avente come estremi* $(1,0)$ *e* $(1,1)$. *È comunque sempre vero che esiste almeno una soluzione del problema (7.3) che è anche soluzione del problema (7.1).*

Dopo aver chiarito cosa sia la chiusura convessa di un insieme e l'importanza di tale chiusura per i problemi di PLI, ci concentreremo ora su speciali sottoclassi di problemi di PLI, quelli per cui la chiusura convessa di $Z_a = S_a \cap Z^n$ coincide con S_a, la regione ammissibile del rilassamento lineare, ovvero

$$conv(Z_a) = conv(S_a \cap Z^n) = S_a. \tag{7.4}$$

Si noti che nell'esempio precedente ciò non era vero. I problemi di PLI per cui si verifica questa condizione sono importanti perchè sono molto più semplici da risolvere rispetto agli altri problemi di PLI. Infatti, essendo $conv(Z_a) = S_a$ e viste le strette relazioni tra le soluzioni dei problemi (7.1) e (7.3), possiamo risolvere tali problemi semplicemente eliminando i vincoli di interezza sulle variabili, ovvero risolvendo il problema di PL

$$\max_{x \in S_a} cx.$$

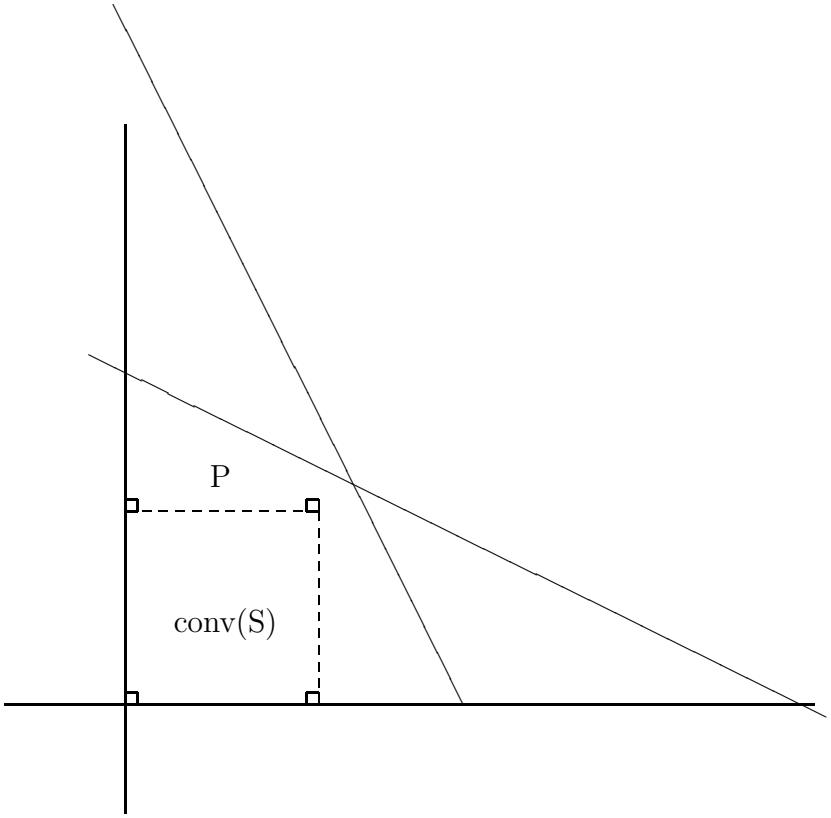


Figura 7.2: La chiusura convessa di Z_a per l'esempio considerato.

Come detto e come chiariremo meglio nel Capitolo 10 i problemi di PL sono molto più semplici da risolvere dei problemi di PLI e quindi le sottoclassi di problemi di PLI che soddisfano la condizione vista sono in generale molto più semplici da risolvere dei generici problemi di PLI. Inoltre, in molti casi questi problemi hanno strutture particolari che consentono di risolverli attraverso algoritmi specifici che sono per lo più specializzazioni di metodi per risolvere problemi di PL (come il metodo del simplesso e le sue varianti), dove la particolare struttura dei problemi consente di implemementare in modo più efficiente i passi di tali metodi.
Nel seguito mostreremo importanti sottoclassi per cui la condizione (7.4) è soddisfatta.

7.2.2 Matrici totalmente unimodulari

Prima di approfondire ulteriormente il discorso sui problemi per cui $conv(Z_a) = S_a$, introduciamo il concetto di matrice *totalmente unimodulare*.

Definizione 2 *Una matrice A si dice totalmente unimodulare (TU nel seguito) se ogni sua sottomatrice quadrata ha determinante pari a 0, +1 o -1.*

Si noti che una matrice TU può avere come elementi solo i valori 0, +1 e -1 visto che ogni suo elemento è una particolare sottomatrice quadrata di ordine 1×1. Si citano di seguito alcune proprietà delle matrici TU.

Proprietá 1 *Se A è una matrice TU si ha che*

1. A^T *è TU*
2. $[A\ I]$*, dove I è la matrice identica, è TU*
3. *una matrice ottenuta duplicando righe e/o colonne di A è ancora TU*
4. *una matrice ottenuta moltiplicando righe e/o colonne di A per -1 è ancora TU*
5. *una matrice ottenuta scambiando righe di A (oppure colonne di A) tra loro è ancora TU*
6. *una matrice ottenuta da A mediante un'operazione di cardine è ancora TU*

Ci chiediamo ora come sia possibile riconoscere una matrice TU senza dover calcolare i determinanti di tutte le sottomatrici quadrate. Esistono alcune regole, tra cui la seguente.

Osservazione 19 *Sia A una matrice i cui elementi sono tutti uguali a 0, +1 o -1 e lungo ogni colonna non vi sono più di due elementi diversi da 0. Allora A è TU se e solo se l'insieme delle righe di A può essere suddiviso in due sottinsiemi* Q_1 *e* Q_2 *tali che se una colonna contiene due elementi diversi da 0 si ha che:*

- *se i due elementi hanno lo stesso segno allora una delle due righe in cui si trovano è in* Q_1 *e l'altra in* Q_2*;*
- *se hanno segno opposto le righe corrispondenti sono entrambe contenute in* Q_1 *od entrambe in* Q_2*.*

Esempio 29 *Sia A la seguente matrice*

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Prendendo $Q_1 = \{1, 2\}$ *e* $Q_2 = \{3, 4\}$ *si verifica immediatamente che la condizione è soddisfatta e quindi A è TU.*

Vale il seguente corollario.

Corollario 1 *Sia A una matrice i cui elementi sono tutti uguali a 0, +1 o -1 e lungo ogni colonna non vi sono più di due elementi diversi da 0. Se nelle colonne con due elementi diversi da 0 la somma di tali elementi è uguale a 0 (ovvero un elemento è uguale a +1 e l'altro a -1), allora A è TU.*

Dimostrazione È sufficiente utilizzare l'Osservazione 19 ponendo

$$Q_1 = \{\text{tutte le righe di } A\} \quad Q_2 = \emptyset.$$

Il corollario ci dice che vale la seguente osservazione.

Osservazione 20 *Tutte le matrici di incidenza nodo-arco di un grafo orientato e anche tutte le matrici ottenute da queste rimuovendone alcune righe sono matrici TU.*

Esempio 30 *Sia A la seguente matrice*

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

In base al corollario tale matrice è TU. Se si rimuove una qualsiasi delle righe la matrice ottenuta continua a essere TU.

Un'altra osseravzione è la seguente:

Osservazione 21 *La matrice di incidenza nodo-arco per un grafo bipartito non orientato $G = (V_1 \cup V_2, A)$, dove V_1 e V_2 sono le due classi di bipartizione, è TU.*

Dimostrazione Si ricordi che per grafi non orientati la matrice di incidenza nodo-arco è una matrice con entrate pari a 0 o 1, con tante righe quanti sono i nodi del grafo e tante colonne quanti sono gli archi del grafo. Lungo la colonna relativa ad un arco $(i, j) \in A$ la matrice ha un +1 all'altezza delle righe i e j e 0 in tutte le altre righe. Per dimostrare il corollario è sufficiente utilizzare l'Osservazione 19 ponendo

$$Q_1 = V_1 \quad Q_2 = V_2.$$

Ma perché sono importanti le matrici TU? La loro importanza è legata a questo teorema (non dimostrato).

Teorema 6 *Sia*

$$S_a(b_1, b_2, b_3) = \{x \in R^n : \ A_1 x \leq b_1, \ A_2 x \geq b_2, \ A_3 x = b_3 \ x \geq 0\}$$

e

$$Z_a(b_1, b_2, b_3) = S_a(b_1, b_2, b_3) \cap Z^n,$$

con b_1, b_2, b_3 *vettori di interi. Si dimostra che* A_1, A_2, A_3 *sono TU se e solo se per tutti i vettori di interi* b_1, b_2, b_3 *per cui* $S_a(b_1, b_2, b_3) \neq \emptyset$ *si ha che*

$$conv(Z_a(b_1, b_2, b_3)) = S_a(b_1, b_2, b_3).$$

La conseguenza principale di questo teorema è che la soluzione di un problema di PLI con matrici dei vincoli TU e vettori dei termini noti interi può essere ottenuta semplicemente eliminando i vincoli di interezza.

Esempio 31 *Si consideri il seguente problema*

$$\begin{aligned}
&\max\ x_1 + x_2 + x_3 + x_4 \\
&x_1 + x_2 = 2 \\
&-x_1 + x_3 = 4 \\
&-x_2 + x_4 = 3 \\
&-x_3 - x_4 = 2 \\
&x_1, x_2, x_3, x_4 \geq 0 \quad \textit{interi}
\end{aligned}$$

Il problema di PL ottenuto eliminando i vincoli di interezza ha regione ammissibile

$$S_a(b_3) = \{x \in R^n :\ A_3 x = b_3,\ x \geq 0\}$$

con

$$A_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

e

$$b_3 = \begin{bmatrix} 2 \\ 4 \\ 3 \\ 2 \end{bmatrix}$$

Si può verificare attraverso il Corollario 1 che A_3 *è TU. Essendo* b_3 *un vettore di interi, il problema di PLI può essere risolto eliminando semplicemente i vincoli di interezza.*

Questi risultati coi consentono ora di dimostrare la seguente osservazione.

Osservazione 22 *I problemi di:*

- *flusso a costo minimo*
- *flusso massimo*
- *trasporto*

- *assegnamento*

sono tutti risolvibili come problemi di PL rimuovendo i vincoli di interezza sulla variabili.

Dimostrazione Le Osservazioni 1-3 e le Osservazioni 20-21 mostrano che le matrici dei vincoli di questi problemi sono tutte TU, mentre per definizione dei problemi tutti i termini noti sono interi.

Capitolo 8

Algoritmi di taglio

In questo capitolo vedremo una prima metodologia di risoluzione per problemi di PLI, gli *algoritmi di taglio*. Prima però di addentrarci nella descrizione di questi facciamo un'osservazione che ci servirà per essi e anche per i successivi algoritmi di branch-and-bound.

8.1 Trasformazione di un problema di PLI in forma standard

Come per i problemi di PL, è sempre possibile ricondurre un problema di PLI in forma standard. Le regole per effettuare questa trasformazione sono le stesse già viste per i problemi di PL ma occorre prestare attenzione ad un ulteriore aspetto. Vediamo di illustrare questo su un semplice esempio. Si consideri il seguente problema di PLI:

$$\begin{array}{ll} \max & x_2 \\ & x_1 \leq \frac{1}{2} \\ & x_2 \leq \frac{1}{2} \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{array}$$

Se avessimo a che fare con un problema di PL potremmo trasformarlo molto semplicemente nella forma standard con l'aggiunta di due variabili y_1 e y_2:

$$\begin{array}{ll} \max & x_2 \\ & x_1 + y_1 = \frac{1}{2} \\ & x_2 + y_2 = \frac{1}{2} \\ & x_1, x_2, y_1, y_2 \geq 0, \quad x_1, x_2 \in Z. \end{array}$$

In questo modo ci ritroveremmo però con un problema in cui alcune variabili (x_1 e x_2) possono assumere solo valori interi e altre possono assumere anche

valori non interi (non avremmo né un problema di PL puro, né un problema di PLI puro). Infatti, ad esempio, se scelgo $x_1 = x_2 = 0$, valori ammissibili per il nostro problema di PLI, il corrispondente valore di y_1 e y_2 è pari a $1/2$. Per ovviare a questo inconveniente e fare in modo che anche le nuove variabili possano assumere solo valori interi quando quelle originarie hanno valori interi, è sufficiente trasformare i vincoli in modo tale che in essi compaiano solo coefficienti e termini noti interi. Nel nostro esempio basta moltiplicare entrambi i vincoli per 2:

$$\begin{aligned} \max \quad & x_2 \\ & 2x_1 \leq 1 \\ & 2x_2 \leq 1 \\ & x_1, x_2 \geq 0, \quad x_1, x_2 \in Z. \end{aligned}$$

e solo a questo punto aggiungere le due variabili y_1 e y_2:

$$\begin{aligned} \max \quad & x_2 \\ & 2x_1 + y_1 = 1 \\ & 2x_2 + y_2 = 1 \\ & x_1, x_2, y_1, y_2 \geq 0, \quad x_1, x_2, y_1, y_2 \in Z. \end{aligned}$$

In questo modo le variabili aggiunte y_1, y_2 assumono sempre valori interi quando quelle originarie x_1, x_2 hanno valori interi e quindi nella forma standard possiamo imporre l'interezza anche delle variabili y_1, y_2. Si noti che il funzionamento dei successivi algoritmi di risoluzione è garantito solo se si procede alla trasformazione in forma standard nel modo sopra indicato.

8.2 Algoritmi di taglio

Per introdurre gli algoritmi di taglio dobbiamo prima introdurre il concetto di taglio valido per un problema di PLI. Sia dato un problema di PLI con il suo rilassamento lineare. Sia $\mathbf{x}^*$ una soluzione ottima del rilassamento lineare, che si suppone abbia almeno una coordinata non intera (se tutte le sue coordinate fossero intere allora $\mathbf{x}^* \in Z_{ott}$).

Definizione 12 *Una disequazione* $\mathbf{wx} \leq v$ *si definisce* taglio valido *per il problema di PLI se non è soddisfatta da* $\mathbf{x}^*$ *ma è soddisfatta da tutti i punti nella regione ammissibile del problema di PLI, ovvero*

$$\mathbf{wx}^* > \mathbf{v}, \qquad \mathbf{wx} \leq \mathbf{v} \quad \forall\ \mathbf{x} \in Z_a$$

Il concetto di taglio valido conduce alla definizione degli *algoritmi di taglio*. In essi si comincia risolvendo il rilassamento lineare del problema di PLI. Se questo ha soluzione a coordinate tutte intere, essa è soluzione anche del problema di PLI. Altrimenti si genera (in modi che vedremo in seguito) un taglio valido, si

aggiunge questo taglio ai vincoli del rilassamento lineare e si risolve il nuovo problema di PL ottenuto con questa aggiunta. Si noti che, per la definizione di taglio valido, l'aggiunta dei diversi tagli *non modifica mai la regione ammissibile del problema di PLI ma restringe via via quella del rilassamento lineare, tagliando via porzioni di essa che contengono le successive soluzioni ottime dei rilassamenti lineari risolti dopo l'introduzione dei tagli.*

Se la soluzione ottima ha coordinate tutte intere, essa è soluzione anche del problema di PLI. Altrimenti si genera un nuovo taglio valido che escluda la soluzione ottima del nuovo problema di PL, si aggiunge anche questo taglio ai vincoli del rilassamento lineare e si risolve il nuovo problema di PL ottenuto con questa ulteriore aggiunta. Si itera questa procedura fino a quando non si ottiene una soluzione intera (o si stabilisce che $Z_a = \emptyset$). Notiamo che qui e nel seguito supporremo sempre che il problema di PLI e il suo rilassamento lineare non abbiano mai obiettivo illimitato. Anche se esistono tecniche per trattare tali casi, nella pratica le variabili intere hanno tipicamente dei limiti che ne impediscono la crescita o decrescita illimitata, escludendo quindi la possibilità di obiettivo illimitato. Dato il problema di PLI

$$\begin{array}{lll} \max & \mathbf{cx} & \\ & \mathbf{a}_i\mathbf{x} = b_i & i = 1, \ldots, m \\ & x_j \geq 0, \quad x_j \in I & j = 1, \ldots, n \end{array}$$

formalmente, lo schema di un algoritmo di taglio è il seguente.

Inizializzazione Si risolva il rilassamento lineare

$$\begin{array}{lll} \max & \mathbf{cx} & \\ & \mathbf{a}_i\mathbf{x} = b_i & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n \end{array}$$

del problema di PLI. Se $S_a = \emptyset$, allora $Z_a = \emptyset$. Altrimenti (tralasciando il caso di obiettivo illimitato) esiste una soluzione ottima, indicata con $\mathbf{x}^{*1}$. Si ponga $k = 1$.

Passo 1 Se $\mathbf{x}^{*k}$ ha coordinate tutte intere, allora si restituisca $\mathbf{x}^{*k} \in Z_{ott}$. Altrimenti si vada al Passo 2.

Passo 2 Si generi un taglio valido $\mathbf{w}_k\mathbf{x} \leq v_k$ che non sia soddisfatto da $\mathbf{x}^{*k}$ ma sia soddisfatto da tutti i punti in Z_a, ovvero

$$\mathbf{w}_k\mathbf{x}^{*k} > v_k, \qquad \mathbf{w}_k\mathbf{x} \leq v_k \quad \forall\, \mathbf{x} \in Z_a.$$

Passo 3 Si aggiunga il nuovo taglio valido ai vincoli originari del problema e

ai tagli validi generati in precedenza e si risolva il problema di PL

$$\begin{array}{lcl} \max & \mathbf{cx} & \\ & \mathbf{a}_i\mathbf{x} = b_i & i = 1, \ldots, m \\ & \mathbf{w}_r\mathbf{x} \leq v_r & r = 1, \ldots, k \\ & x_j \geq 0 & j = 1, \ldots, n \end{array} \tag{8.1}$$

Se tale problema ha regione ammissibile vuota possiamo concludere che $Z_a = \emptyset$. Altrimenti sia $\mathbf{x}^{*(k+1)}$ la sua soluzione ottima.

Passo 4 Si ponga $k = k + 1$ e si ritorni al Passo 1.

Si noti che il problema di PL (8.1) non è in forma standard. Tuttavia basta la semplice aggiunta di una variabile $y_r \geq 0$ in ciascuno dei vincoli $\mathbf{w}_r\mathbf{x} \leq v_r$ per ricondursi al formato standard. Con tale aggiunta il problema di PL (8.1) viene trasformato in quello equivalente (in forma standard):

$$\begin{array}{lcl} \max & \mathbf{cx} & \\ & \mathbf{a}_i\mathbf{x} = b_i & i = 1, \ldots, m \\ & \mathbf{w}_r\mathbf{x} + y_r = v_r & r = 1, \ldots, k \\ & x_j \geq 0 & j = 1, \ldots, n \\ & y_r \geq 0 & r = 1, \ldots, k \end{array} \tag{8.2}$$

Resta ancora da chiarire come si genera un taglio valido. Ci sono molti modi per generare tagli validi. Qui descriveremo un tipo di tagli detti *tagli di Gomory*.

8.2.1 Tagli di Gomory

Supponiamo di avere una base ottima $B^* = \{x_{i_1}, \ldots, x_{i_m}\}$ per il rilassamento lineare del problema di PLI. La riformulazione rispetto a tale base è la seguente:

$$\begin{array}{lc} \max & \gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}} \\ & x_{i_1} = \beta_1 + \sum_{j=1}^{n-m} \alpha_{1j} x_{i_{m+j}} \\ & \cdots \\ & x_{i_k} = \beta_k + \sum_{j=1}^{n-m} \alpha_{kj} x_{i_{m+j}} \\ & \cdots \\ & x_{i_m} = \beta_m + \sum_{j=1}^{n-m} \alpha_{mj} x_{i_{m+j}} \\ & x_1, \ldots, x_n \geq 0 \end{array}$$

Si suppone che almeno uno dei valori β_r, $r = 1, \ldots, m$, sia non intero (se fossero tutti interi la soluzione di base associata a B^* sarebbe non solo ottima per il rilassamento lineare ma anche per il problema di PLI). Sia β_k un tale valore non intero. Scriviamo l'equazione relativa a x_{i_k}:

$$x_{i_k} = \beta_k + \alpha_{k1} x_{i_{m+1}} + \alpha_{k2} x_{i_{m+2}} + \cdots + \alpha_{k,n-m} x_{i_n}. \tag{8.3}$$

Si consideri ora il seguente taglio, detto *taglio di Gomory*:

$$-f_k + f_{k1}x_{i_{m+1}} + f_{k2}x_{i_{m+2}} + \cdots + f_{k,n-m}x_{i_n} \geq 0$$

dove f_{kj}, $j = 1, \ldots, n-m$, è la mantissa di $-\alpha_{kj}$, cioè

$$f_{kj} = -\alpha_{kj} - \lfloor -\alpha_{kj} \rfloor \geq 0$$

($\lfloor a \rfloor$ rappresenta la parte intera di a ovvero il più grande intero $\leq a$), mentre f_k è la mantissa di β_k, cioè

$$f_k = \beta_k - \lfloor \beta_k \rfloor > 0,$$

dove $f_k > 0$ è una conseguenza della non interezza di β_k. Per mantenere il formato standard, possiamo aggiungere una nuova variabile y_1 e riscrivere il taglio attraverso la seguente coppia di vincoli:

$$y_1 = -f_k + f_{k1}x_{i_{m+1}} + f_{k2}x_{i_{m+2}} + \cdots + f_{k,n-m}x_{i_n} \tag{8.4}$$

$$y_1 \geq 0.$$

Vogliamo dimostrare la seguente osservazione.

Osservazione 23 *Il taglio di Gomory è un taglio valido.*

Dimostrazione Per prima cosa dimostriamo che la soluzione ottima del rilassamento lineare non soddisfa questo vincolo. Notiamo che per tale soluzione ottima si ha $x_{i_{m+1}} = \cdots = x_{i_n} = 0$ (tutte le variabili fuori base sono nulle) e quindi, in corrispondenza della soluzione ottima del rilassamento lineare si ha:

$$y_1 = -f_k < 0.$$

Quindi la soluzione ottima del rilassamento lineare non soddisfa, come desiderato, il nostro taglio. Resta da far vedere che il taglio è soddisfatto da ogni punto in Z_a. Prendiamo un generico punto

$$(\overline{x}_{i_1}, \ldots, \overline{x}_{i_n})$$

in Z_a. Sostituiamo le coordinate di tale punto in (8.3) e (8.4) ed otteniamo rispettivamente

$$\overline{x}_{i_k} = \beta_k + \alpha_{k1}\overline{x}_{i_{m+1}} + \alpha_{k2}\overline{x}_{i_{m+2}} + \cdots + \alpha_{k,n-m}\overline{x}_{i_n}. \tag{8.5}$$

$$\overline{y}_1 = -f_k + f_{k1}\overline{x}_{i_{m+1}} + f_{k2}\overline{x}_{i_{m+2}} + \cdots + f_{k,n-m}\overline{x}_{i_n}. \tag{8.6}$$

Ciò che si vuole dimostrare è che il valore di $\overline{y}_1$ è ≥ 0 e cioè che la generica soluzione ammissibile in Z_a

$$\overline{x}_{i_1}, \ldots, \overline{x}_{i_n} \tag{8.7}$$

soddisfa il taglio. Per prima cosa dimostriamo che in corrispondenza di ogni punto (8.7) in Z_a si ha che il valore di $\overline{y}_1$ è intero. Per dimostrare questo

sommiamo (8.5) e (8.6). Ricordando la definizione delle f_{kj} e di f_k, la somma ha come risultato

$$\overline{y}_1 = \lfloor \beta_k \rfloor - \overline{x}_{i_k} - \lfloor -\alpha_{k1} \rfloor \overline{x}_{i_{m+1}} - \lfloor -\alpha_{k2} \rfloor \overline{x}_{i_{m+2}} - \cdots - \lfloor -\alpha_{k,n-m} \rfloor \overline{x}_{i_n}$$

Questo mostra che in corrispondenza di ogni punto in Z_a il valore $\overline{y}_1$ è un valore intero (le parti intere sono tutti valori interi e le variabili $\overline{x}_{i_k}$ e $\overline{x}_{i_{m+j}}$, $j = 1, \ldots, n-m$ in Z_a assumono valori interi).
Da (8.6) abbiamo anche che

$$\overline{y}_1 + f_k = \underbrace{f_{k1}}_{\geq 0} \underbrace{\overline{x}_{i_{m+1}}}_{\geq 0} + \underbrace{f_{k2}}_{\geq 0} \underbrace{\overline{x}_{i_{m+2}}}_{\geq 0} + \cdots + \underbrace{f_{k,n-m}}_{\geq 0} \underbrace{\overline{x}_{i_n}}_{\geq 0}.$$

Questo ci dice che, in corrispondenza di ogni punto (8.7) di Z_a, si ha che $\overline{y}_1 + f_k$ è ≥ 0 ed inoltre sappiamo che $0 < f_k < 1$. Infine, in precedenza abbiamo dimostrato che $\overline{y}_1$ assume valori interi in corrispondenza di ogni punto di Z_a. Tutto questo è possibile soltanto se $\overline{y}_1 \geq 0$ in corrispondenza di ogni punto di Z_a, come si voleva dimostrare.

La coppia di vincoli

$$y_1 = -f_k + f_{k1} x_{i_{m+1}} + f_{k2} x_{i_{m+2}} + \cdots + f_{k,n-m} x_{i_n}$$

$$y_1 \geq 0.$$

che esprime il taglio viene aggiunta alla riformulazione rispetto alla base ottima B^* e si ottiene quindi:

$$\begin{array}{ll} \max & \gamma_0 + \sum_{j=1}^{n-m} \gamma_j x_{i_{m+j}} \\ & x_{i_1} = \beta_1 + \sum_{j=1}^{n-m} \alpha_{1j} x_{i_{m+j}} \\ & \cdots \\ & x_{i_k} = \beta_k + \sum_{j=1}^{n-m} \alpha_{kj} x_{i_{m+j}} \\ & \cdots \\ & x_{i_m} = \beta_m + \sum_{j=1}^{n-m} \alpha_{mj} x_{i_{m+j}} \\ & y_1 = -f_k + f_{k1} x_{i_{m+1}} + f_{k2} x_{i_{m+2}} + \cdots + f_{k,n-m} x_{i_n} \\ & x_1, \ldots, x_n, y_1 \geq 0 \end{array} \tag{8.8}$$

Si noti come (8.8) sia già la riformulazione del nuovo problema di PL rispetto alla base $B^* \cup \{y_1\}$. La soluzione di base del primale associata a questa base è non ammissibile (la variabile y_1 ha valore $-f_k < 0$), come deve essere visto che il taglio deve rendere non ammissibile la soluzione del rilassamento lineare. Tuttavia la soluzione di base del duale associata a $B^* \cup \{y_1\}$ è ammissibile (i valori γ_j continuano ad essere non positivi visto che non sono stati modoficati dall'introduzione del taglio). Questo ci consente di risolvere il nuovo problema di PL con il simplesso duale e con base iniziale $B^* \cup \{y_1\}$. Si avrà anche che la prima variabile ad uscire di base sarà la y_1. Perché?

Si noti anche come durante la dimostrazione dell'Osservazione 23 abbiamo anche dimostrato che la nuova variabile che viene introdotta (la y_1) assume sempre valori interi in corrispondenza di ogni punto di Z_a. In pratica il risultato di interezza su y_1 ci consente di dire che (8.8) con l'ulteriore imposizione

$$x_1, \ldots, x_n, y_1 \in Z$$

è una riformulazione equivalente del problema di PLI iniziale (dove per equivalente si intende che Z_a e Z_{ott} restano invariati). Più in generale, iterando quanto visto, si ha che (8.2) con l'ulteriore imposizione

$$x_1, \ldots, x_n, y_1, \ldots, y_k \in Z$$

è sempre una riformulazione equivalente del problema di PLI iniziale. Questo consente di iterare la procedura, cioè se dopo l'aggiunta del primo taglio e la risoluzione del nuovo problema di PL (8.8) non si ha ancora una soluzione intera, possiamo generare un nuovo taglio utilizzando la stessa regola di generazione.
Ma vediamo ora di chiarire meglio il funzionamento di un algoritmo di taglio che utilizza tagli di Gomory applicandolo a un problema di esempio. Si consideri il seguente problema di PLI:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & 2x_1 + x_3 = 3 \\ & 2x_2 + x_4 = 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2, x_3, x_4 \in Z. \end{aligned}$$

Una base otttima per il rilassamento lineare di questo problema di PLI è la base $B^* = \{x_1, x_2\}$. La riformulazione del rilassamento lineare rispetto a questa base è la seguente:

$$\begin{aligned} \max \quad & 3 - 1/2x_3 - 1/2x_4 \\ & x_1 = 3/2 - 1/2x_3 \\ & x_2 = 3/2 - 1/2x_4 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \tag{8.9}$$

Come si vede c'è un termine noto non intero quindi la soluzione del rilassamento lineare:

$$x_1^* = x_2^* = 3/2 \quad x_3^* = x_4^* = 0$$

non risolve il problema di PLI. A questo punto aggiungo un taglio di Gomory. Considero l'equazione

$$x_1 = 3/2 - 1/2x_3$$

e da questa, in base alle regole viste, si ricava il seguente taglio:

$$-1/2 + 1/2x_3 \geq 0.$$

Aggiungendo una variabile y_1 il taglio viene espresso dalla seguente coppia di vincoli:

$$y_1 = -1/2 + 1/2x_3, \quad y_1 \geq 0.$$

Aggiungiamo ora questi alla riformulazione (8.9) rispetto alla base ottima B^*.

$$\begin{aligned} \max \quad & 3 - 1/2x_3 - 1/2x_4 \\ & x_1 = 3/2 - 1/2x_3 \\ & x_2 = 3/2 - 1/2x_4 \\ & y_1 = -1/2 + 1/2x_3 \\ & x_1, x_2, x_3, x_4, y_1 \geq 0 \end{aligned}$$

Possiamo notare che questa è già la riformulazione del nuovo problema rispetto alla base $B^* \cup \{y_1\} = \{x_1, x_2, y_1\}$. La soluzione di base del primale associata a questa base è non ammissibile (il valore di y_1 è $-1/2$) ma la soluzione di base del duale associata alla stessa base è ammissibile (i coefficienti delle variabili fuori base nell'obiettivo non sono stati modificati e continuano ad essere negativi). Quindi possiamo applicare il simplesso duale con base iniziale $\{x_1, x_2, y_1\}$. Si verifica immediatamente che non sono soddisfatte né la condizione di ottimalità, né la condizione di illimitatezza dell'obiettivo duale. Chiaramente, la prima variabile che dovrà entrare in base sarà l'unica con un valore negativo e quindi dovrà essere la y_1. In base alle regole viste, l'unica variabile che potrà entrare in base è la x_3. Passeremo quindi alla base $\{x_1, x_2, x_3\}$ rispetto alla quale avremo la seguente riformulazione:

$$\begin{aligned} \max \quad & 5/2 - y_1 - 1/2x_4 \\ & x_1 = 1 - y_1 \\ & x_2 = 3/2 - 1/2x_4 \\ & x_3 = 1 + 2y_1 \\ & x_1, x_2, x_3, x_4, y_1 \geq 0 \end{aligned} \tag{8.10}$$

La base è ottima per il nuovo problema di PL ma purtroppo abbiamo ancora un termine noto non intero e quindi non abbiamo ancora una soluzione ottima del problema di PLI. Allora, dobbiamo aggiungere un ulteriore taglio di Gomory. La regola di generazione è la solita. Per prima cosa selezioniamo un'equazione con termine noto non intero. Qui la sola possibile è:

$$x_2 = 3/2 - 1/2x_4.$$

Da questa si genera, in base alle regole viste, il seguente taglio:

$$-1/2 + 1/2x_4 \geq 0.$$

Aggiungendo una nuova variabile y_2 il taglio viene espresso dalla seguente coppia di vincoli:

$$y_2 = -1/2 + 1/2x_4, \quad y_2 \geq 0.$$

Aggiungiamo ora questi alla riformulazione (8.10) rispetto alla base ottima $\{x_1, x_2, x_3\}$.

$$\begin{aligned} \max \quad & 5/2 - y_1 - 1/2x_4 \\ & x_1 = 1 - y_1 \\ & x_2 = 3/2 - 1/2x_4 \\ & x_3 = 1 + 2y_1 \\ & y_2 = -1/2 + 1/2x_4 \\ & x_1, x_2, x_3, x_4, y_1, y_2 \geq 0 \end{aligned}$$

Possiamo notare che questa è già la riformulazione del nuovo problema rispetto alla base $\{x_1, x_2, x_3, y_2\}$. La soluzione di base del primale associata a questa base è non ammissibile (il valore di y_2 è $-1/2$) ma la soluzione di base del duale associata alla stessa base è ammissibile (i coefficienti delle variabili fuori base nell'obiettivo non sono stati modificati e continuano ad essere negativi). Quindi possiamo applicare il simplesso duale con base iniziale $\{x_1, x_2, x_3, y_2\}$. Si verifica immediatamente che non sono soddisfatte né la condizione di ottimalità, né la condizione di illimitatezza dell'obiettivo duale. Chiaramente, la prima variabile che dovrà entrare in base sarà l'unica con un valore negativo e quindi dovrà essere la y_2. In base alle regole viste, l'unica variabile che potrà entrare in base è la x_4. Passeremo quindi alla base $\{x_1, x_2, x_3, x_4\}$ rispetto alla quale avremo la seguente riformulazione:

$$\begin{aligned} \max \quad & 2 - y_1 - y_2 \\ & x_1 = 1 - y_1 \\ & x_2 = 1 - y_2 \\ & x_3 = 1 + 2y_1 \\ & x_4 = 1 + 2y_2 \\ & x_1, x_2, x_3, x_4, y_1, y_2 \geq 0 \end{aligned}$$

La base è ottima per il nuovo problema di PL e tutte le variabili hanno valore intero. Abbiamo quindi la seguente soluzione ottima del problema di PLI:

$$x_1^* = x_2^* = x_3^* = x_4^* = 1.$$

Ci si può chiedere se siamo stati solo fortunati oppure con i tagli di Gomory l'algoritmo di taglio termina sempre in un numero finito di iterazioni. Vale la seguente osservazione.

Osservazione 24 *Se ad ogni iterazione il taglio di Gomory viene realizzato a partire dalla prima equazione con un termine noto β_k non intero, allora l'algoritmo termina in un numero finito di iterazioni.*

Capitolo 9

L'approccio branch-and-bound

In questo capitolo daremo una descrizione di un algoritmo branch-and-bound per problemi di ottimizzazione in forma generica (quindi non necessariamente soltanto per problemi di PLI). Una definizione formale di problema di ottimizzazione verrà data nel Capitolo 10. Di seguito vedremo la specializzazione per problemi di PLI. Analizzeremo una per una le componenti principali di un algoritmo branch-and-bound per un problema di massimizzazione generico

$$\max_{x \in S} f(x)$$

con regione ammissibile S e funzione obiettivo f (nei problemi di PLI le abbiamo indicate rispettivamente con Z_a e $\mathbf{cx}$). Verrano quindi segnalate le piccole variazioni che vanno introdotte per problemi di minimo.

9.1 Componenti di un algoritmo branch-and-bound

9.1.1 Calcolo di un upper bound

Data la regione ammissibile S dell'istanza di un problema di massimizzazione, supponiamo di avere un sottinsieme $T \subseteq S$. Una limitazione superiore o *upper bound* per T è un valore $U(T)$ con la seguente proprietà

$$U(T) \geq f(x) \quad \forall\, x \in T.$$

Il valore $U(T)$ viene calcolato tramite una procedura che deve avere come proprietà quella di poter essere eseguita in tempi brevi (in particolare, il calcolo degli upper bound deve richiedere un tempo molto inferiore rispetto al tempo necessario per risolvere l'intero problema). Spesso la scelta di una procedura per il calcolo del'upper bound è fortemente legata al particolare problema che si sta risolvendo. Inoltre, non esiste un'unica procedura per un dato problema.

Un modo ampiamente utilizzato per determinare un upper bound $U(T)$ (o un lower bound $L(T)$ per problemi di minimo), è quello di determinare la soluzione di un suo rilassamento. Indichiamo con:

$$\alpha(f, T) = \max_{x \in T} f(x), \tag{9.1}$$

il valore ottimo della funzione f sull'insieme T. Si definisce rilassamento del problema (9.1), un problema:

$$\alpha(f', T') = \max_{x \in T'} f'(x) \tag{9.2}$$

dove:

$$T \subseteq T', \tag{9.3}$$

e

$$f'(x) \geq f(x) \quad \forall\ x \in T \tag{9.4}$$

(per problemi di minimo si deve invertire il verso della diseguaglianza). Vale la seguente osservazione che dimostra che il valore ottimo del rilassamento (9.2) sia un upper bound $U(T)$ per il valore ottimo del problema (9.1).

Osservazione 25 *Si ha che:*

$$\alpha(f', T') \geq \alpha(f, T).$$

Dimostrazione Sia $x^* \in T$ una soluzione ottima del problema (9.1), cioè:

$$f(x^*) = \alpha(f, T),$$

e sia $x' \in T'$ una soluzione ottima del problema (9.2), cioè:

$$f(x') = \alpha(f', T').$$

A causa di (9.3) si ha che $x^* \in T$ implica $x^* \in T'$. Inoltre, come conseguenza di (9.4), si ha:

$$f'(x^*) \geq f(x^*).$$

Infine, l'ottimalità di x' implica $f'(x') \geq f'(x^*)$ e quindi:

$$\alpha(f', T') = f'(x') \geq f'(x^*) \geq f(x^*) = \alpha(f, T),$$

come si voleva dimostrare.

Il calcolo dell'upper bound tramite la risoluzione del rilassamento (9.2) del problema (9.1) deve avere la proprietà di essere effettuabile in tempi molto più rapidi rispetto al problema (9.1). Esistono molti possibili rilassamenti di un problema. Tra questi, nel caso di problemi di PLI, uno che è già stato incontrato è il *rilassamento lineare*. Dato il generico problema di PLI:

$$\begin{aligned} \max \quad & cx \\ & Ax \leq b \\ & x \geq 0 \quad x \in Z^n, \end{aligned} \tag{9.5}$$

sappiamo che questo è un particolare problema di ottimizzazione con:

$$f(x) = cx \quad T = \{x \in Z^n \ : \ Ax \leq b, \ x \geq 0\}.$$

Il rilassamento lineare di tale problema è un problema della forma (9.2) con:

$$f'(x) \equiv f(x) \quad T' = \{x \in R^n \ : \ Ax \leq b, \ x \geq 0\},$$

dove si usa la stessa funzione obiettivo (il che rende banalmente vero (9.4)) ma nella regione ammissibile T' si accettano anche gli eventuali punti a coordinate non intere oltre a quelli in T e quindi la condizione (9.3) è soddisfatta. Quindi il rilassamento lineare coincide con il seguente problema di PL:

$$\begin{array}{ll} \max & cx \\ & Ax \leq b \\ & x \geq 0 \end{array} \tag{9.6}$$

Si noti che, come richiesto, il rilassamento lineare, essendo un problema di PL, è risolvibile in tempi molto più rapidi dell'originario problema (9.5) di PLI.

Un altro possibile rilassamento, sempre per problemi di PLI,è il *rilassamento lagrangiano*. Supponiamo che il nostro problema sia formulato come segue:

$$\begin{array}{ll} \max & cx \\ & Ax \leq b \\ & Cx \leq d \\ & x \geq 0 \quad x \in Z^n. \end{array}$$

Quindi con:

$$f(x) = cx, \quad T = \{x \in Z^n \ : \ Ax \leq b, \ Cx \leq d, \ x \geq 0\}.$$

Supponiamo che i vincoli $Ax \leq b$ siano facili (ad esempio, A è TU e b è a coordinate tutte intere). Quindi eliminando i vincoli difficili $Cx \leq d$ resta un problema di PLI facile da risolvere (basta risolverne il rilassamento lineare). Per eliminarli li spostiamo nell'obiettivo. Dato un vettore $\lambda \geq 0$, detto *vettore dei moltiplicatori di Lagrange*, delle stesse dimensioni di d, il rilassamento lagrangiano è il seguente:

$$\begin{array}{lll} u(\lambda) = \max & cx + \lambda(d - Cx) & \\ & Ax \leq b & \\ & x \geq 0 & x \in Z^n. \end{array}$$

con

$$f'(x) = cx + \lambda(d - Cx)$$

e

$$T' = \{x \in Z^n \ : \ Ax \leq b, \ x \geq 0\}.$$

Ovviamente, $T \subseteq T'$. Inoltre, per ogni $x \in T$ si ha che:

$$Cx \leq d \;\Rightarrow\; \forall\, \lambda \geq 0 :\; \lambda(d - Cx) \geq 0 \;\Rightarrow\; f'(x) \geq f(x).$$

Quindi sono soddisfatte le due condizioni che devono essere soddisfatte da un rilassamento. Notiamo infine che nel rilassamento lagrangiano rimangono solo i vincoli facili e quindi esso può essere risolto molto più facilmente del problema originario, come viene richiesto per il calcolo di un upper bound. Notiamo anche che ad ogni $\lambda \geq 0$ distinto corisponde un diverso upper bound $u(\lambda)$. Per ottenere il miglior upper bound possibile (ovvero il più piccolo), possiamo risolvere questo ulteriore problema:

$$\min_{\lambda \geq 0} u(\lambda)$$

detto *duale lagrangiano*.

Un caso particolare di rilassamento lagrangiano si ha prendendo tutti i moltiplicatori di Lagrange nulli, cioè ponendo $\lambda = 0$. Questo coincide con il rilassamento ottenuto semplicemente omettendo dal problema i vincoli difficili.

Osserviamo anche che in alcuni casi i vincoli difficili del problema sono vincoli di uguaglianza

$$Cx = d.$$

In tal caso, il rilassamento lagrangiano si definisce nello stesso modo ma i moltiplicatori di Lagrange relativi ai vincoli di uguaglianza non sono vincolati ad assumere solo valori non negativi e possono assumere anche valori negativi.

9.1.2 Calcolo del lower bound

Vogliamo ora calcolare un limite inferiore o *lower bound* per il valore ottimo del nostro problema, ovvero un valore LB con la proprietà che

$$LB \leq f(x^*) = \max_{x \in S} f(x).$$

Si può notare che se prendiamo un qualsiasi elemento $\overline{x} \in S$ e valutiamo in esso la funzione f, il valore $f(\overline{x})$ è già un lower bound, dal momento che $f(\overline{x}) \leq f(x^*)$. Durante l'esecuzione di un algoritmo branch-and-bound la funzione f viene valutata per molti elementi $y_1, \ldots, y_h \in S$ e per ognuno di essi si ha

$$f(y_i) \leq f(x^*) \qquad i = 1, \ldots, h.$$

Quindi, come lower bound possiamo prendere il massimo dei valori f per tali elementi, cioè:

$$LB = \max\{f(y_i) \;:\; i = 1, \ldots, h\} \leq f(x^*).$$

Resta da chiarire da dove ricaviamo gli elementi di S in cui valutare la funzione f durante l'esecuzione dell'algoritmo. Se si ha a disposizione un'euristica (si veda il Capitolo 15) è buona norma valutare f nel risultato di tale euristica.

Inoltre, durante lo stesso calcolo degli upper bound si possono individuare uno o più elementi di S e valutare in essi f. Ad esempio, se si calcola l'upper bound $U(T)$ tramite un rilassamento, nei casi in cui per la soluzione $x' \in T' \supseteq T$ valga anche $x' \in T$, allora si ha anche $x' \in S$ e si può valutare f in x'. In altri casi non si ha $x' \in T$ ma con opportune operazioni (quali arrotondamenti o approssimazioni per eccesso/difetto di valori di variabili) si può determinare partendo da $x' \notin T$ una soluzione $\overline{x}' \in T$.

9.1.3 Branching

L'operazione di branching consiste nel rimpiazzare un insieme $T \subseteq S$ con una sua partizione $T_1, \ldots, T_m$. Si ricordi che $T_1, \ldots, T_m$ formano una partizione di T se

$$T = \cup_{i=1}^{m} T_i \quad T_i \cap T_j = \emptyset \ \ \forall\, i \neq j.$$

La partizione può essere rappresentata tramite una struttura ad albero come in Figura 9.1: l'insieme T è un nodo dell'albero da cui partono i rami (da qui il nome branching) verso i nodi della partizione, che vengono anche detti nodi successori o nodi figli del nodo T.

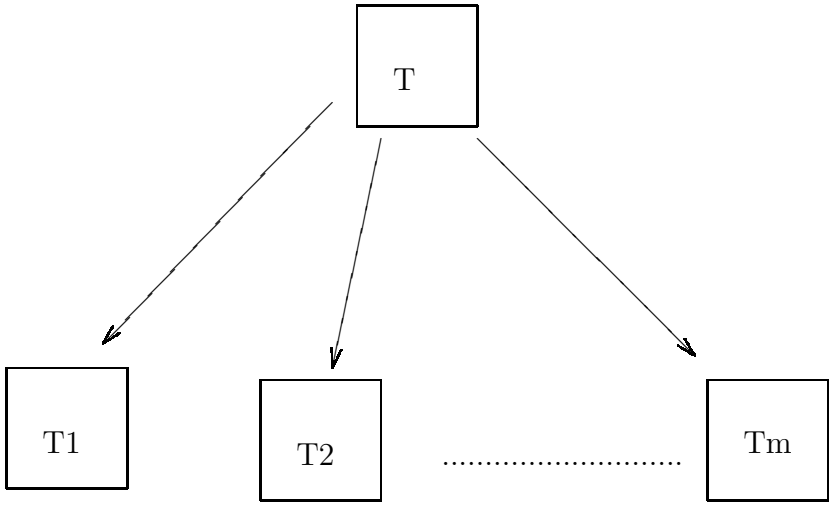


Figura 9.1: Il branching di un nodo T.

9.1.4 Cancellazione di sottinsiemi

Veniamo ora al punto chiave degli algoritmi di branch-and-bound, ovvero la cancellazione di sottinsiemi. Supponiamo che per un dato sottinsieme, T_2 ad esempio, si abbia

$$U(T_2) \leq LB.$$

Ma questo vuol dire che

$$\forall\, x \in T_2 \quad f(x) \leq U(T_2) \leq LB,$$

e cioè tra tutti gli elementi in T_2 non ne possiamo trovare alcuno con valore di f superiore a LB, ovvero al miglior valore di f osservato fino a questo momento. A questo punto posso *cancellare* il sottinsieme T_2 (si veda la Figura 9.2). In

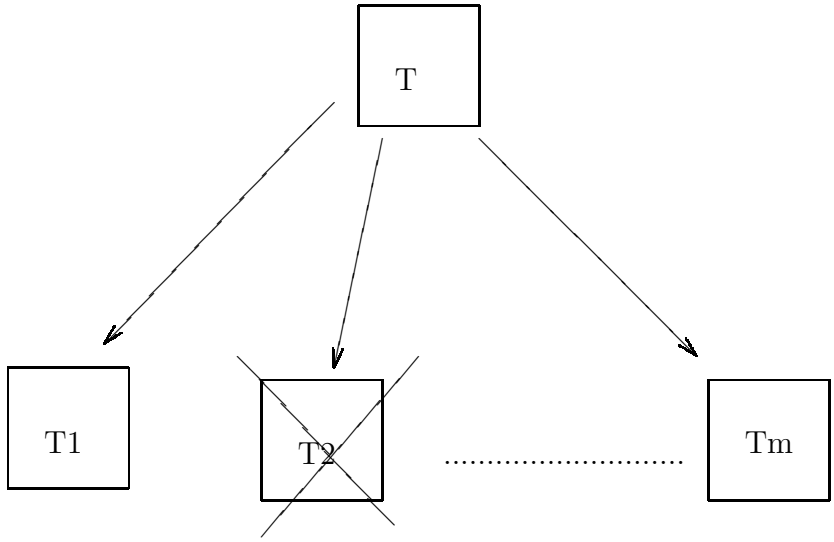


Figura 9.2: La cancellazione del nodo T_2.

questo senso si parla di *enumerazione implicita*: il confronto tra upper bound $U(T_2)$ del sottinsieme e lower bound LB ci consente di scartare tutti gli elementi in T_2 *senza dover calcolare esplicitamente la funzione* f *in essi*.

9.2 L'algoritmo branch-and-bound

Siamo ora pronti per mettere insieme le componenti descritte sopra e formulare il generico algoritmo di branch-and-bound.

Passo 1 Si ponga $\mathcal{C} = \{S\}$ e $\mathcal{Q} = \emptyset$ (l'insieme $\mathcal{C}$ conterrà sempre i sottinsiemi ancora da tenere in considerazione e inizialmente contiene l'intero insieme S, mentre l'insieme $\mathcal{Q}$, inizialmente vuoto, conterrà tutti i sottinsiemi cancellati). Si ponga $k = 1$. Si calcoli $U(S)$ e si calcoli un valore per LB (eventualmente utilizzando anche i risultati di un'euristica, se disponibile). Se non si hanno a disposizione soluzioni in S, allora si ponga $LB = -\infty$.

Passo 2 (Selezione di un sottinsieme) Si selezioni un sottinsieme $T \in \mathcal{C}$. Tra le varie regole di selezione citiamo qui quella di selezionare il sottin-

sieme T in $\mathcal{C}$ con il valore di upper bound più elevato, cioè

$$U(T) = \max_{Q \in \mathcal{C}} U(Q).$$

Passo 3 (Branching) Si sostituisca l'insieme T in $\mathcal{C}$ con la sua partizione in m_k sottinsiemi $T_1, \ldots, T_{m_k}$, ovvero

$$\mathcal{C} = \mathcal{C} \cup \{T_1, \ldots, T_{m_k}\} \setminus \{T\}.$$

Passo 4 (Upper bounding) Si calcoli un upper bound $U(T_i)$, $i = 1, \ldots, m_k$ per ogni sottinsieme della partizione.

Passo 5 (Lower bounding) Si aggiorni, eventualmente, il valore LB se i calcoli al Passo 4 hanno individuato punti in S in cui si è valutata la funzione obiettivo (si ricordi che il valore LB corrisponde sempre al massimo dei valori di f osservati durante l'esecuzione dell'algoritmo).

Passo 6 (Cancellazione sottinsiemi) Si escludano da $\mathcal{C}$ tutti i sottinsiemi Q per cui $U(Q) \leq LB$, ovvero

$$\mathcal{C} = \mathcal{C} \setminus \{Q : \ U(Q) \leq LB\}.$$

e si trasferiscano tali sottinsiemi in $\mathcal{Q}$, cioè:

$$\mathcal{Q} = \mathcal{Q} \cup \{Q : \ U(Q) \leq LB\}.$$

Passo 7 Se $\mathcal{C} = \emptyset$: stop, il valore LB coincide con il valore ottimo $f(x^*)$. Altrimenti si ponga $k = k + 1$ e si ritorni al Passo 2.

Va fatto un breve commento circa il Passo 7. Questo dice che se $\mathcal{C} = \emptyset$ si ha che LB è il valore ottimo del nostro problema. Questa affermazione è una conseguenza del fatto che, nel momento in cui $\mathcal{C} = \emptyset$, tutti i sottinsiemi cancellati fino a quel momento, cioè la collezione $\mathcal{Q}$ di sottinsiemi, formano una *partizione dell'intero insieme* S (più in generale, a ogni iterazione sono i sottinsiemi in $\mathcal{C} \cup \mathcal{Q}$ a formare una partizione di S). Quindi tra di essi ve ne è certamente uno, indicato con $T^* \in \mathcal{Q}$, che contiene x^*. Ma poiché T^* è stato cancellato si dovrà avere

$$f(x^*) \leq U(T^*) \leq LB \leq f(x^*),$$

da cui segue immediatamente che $LB = f(x^*)$.

Infine, dobbiamo brevemente commentare le modifiche da apportare per trattare problemi di minimo. In tali problemi si dovranno semplicemente invertire i ruoli di upper e lower bound: a un sottinsieme $Q \subseteq S$ dovrà essere associato un valore di lower bound $L(Q)$; al posto del valore LB avremo un valore UB con la proprietà

$$UB \geq f(x^*) = \min_{x \in S} f(x).$$

Il valore UB sarà il minimo tra i valori osservati della funzione obiettivo in punti della regione ammissibile S. Il sottinsieme Q viene cancellato se è vero che $L(Q) \geq UB$. Al Passo 2 della procedura di branch-and-bound si seleziona un nodo con lower bound più piccolo, ovvero un nodo T tale che

$$L(T) = \max_{Q \in \mathcal{C}} L(Q).$$

9.3 Approccio Branch-and-Bound per la Programmazione Lineare Intera

Illustreremo dapprima l'approccio proposto per generici problemi di PLI attraverso un esempio studiato per via grafica.

9.3.1 Branch-and-Bound: un esempio

Si consideri il seguente problema di PLI che verrà indicato nel seguito come problema P_0

$$\begin{array}{lllll} P_0 : & \max & x_1 + 3x_2 & & \\ & & (u_1) & x_1 \geq \frac{1}{2} & \\ & & (u_2) & x_1 + \frac{7}{5}x_2 \leq \frac{13}{2} & \\ & & (u_3) & -5x_1 + 3x_2 \leq 5 & \\ & & & x_1, x_2 \geq 0 & \\ & & & x_1, x_2 \in I & \end{array}$$

Vediamo di risolvere il rilassamento lineare di P_0, indicato con P_0', ovvero il problema ottenuto rimuovendo da P_0 i vincoli di interezza sulle variabili x_1 e x_2

$$\begin{array}{lllll} P_0' : & \max & x_1 + 3x_2 & & \\ & & (u_1) & x_1 \geq \frac{1}{2} & \\ & & (u_2) & x_1 + \frac{7}{5}x_2 \leq \frac{13}{2} & \\ & & (u_3) & -5x_1 + 3x_2 \leq 5 & \\ & & & x_1, x_2 \geq 0 & \end{array}$$

Essendo il problema in due sole variabili, la risoluzione viene fatta graficamente. Per problemi con più di due variabili si ricorre all'algoritmo del simplesso. Dalla Figura 9.3 si vede che la regione ammissibile di P_0' è rappresentata dal politopo $ABCD$ i cui vertici sono i punti

$$A\ \left(\frac{5}{4}, \frac{15}{4}\right) \quad B\ \left(\frac{1}{2}, \frac{5}{2}\right) \quad C\ \left(\frac{1}{2}, 0\right) \quad D\ \left(\frac{13}{2}, 0\right)$$

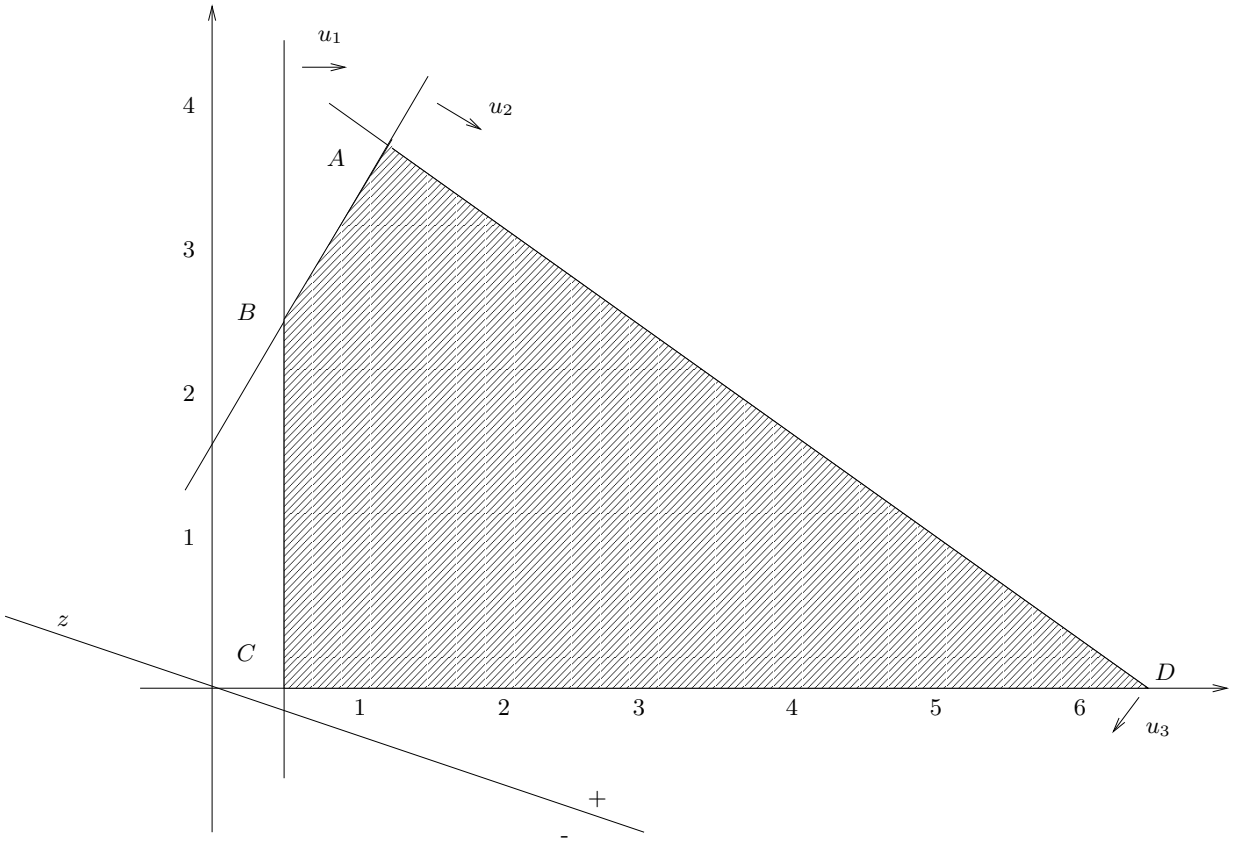


Figura 9.3: Il rilassamento lineare P_0' di P_0, con regione ammissibile $ABCD$.

È immediato verificare che l'unica soluzione ottima di P_0' è il vertice A dove la funzione obiettivo ha valore pari a $\frac{50}{4}$. Se il vertice A avesse coordinate intere sarebbe anche un punto di ottimo per il problema P_0, ma nel nostro esempio ciò non accade. Fino a questo punto non abbiamo sostanziali differenze rispetto agli algoritmi di taglio visti in precedenza. A questo punto un algoritmo di taglio procederebbe introducendo un nuovo vincolo che non è soddifatto dal vertice ottimo di P_0', il vertice A, ma è soddisfatto da tutte le soluzioni ammissibili di P_0. Qui però adottiamo un approccio diverso. Suddividiamo il problema P_0 in due sottoproblemi P_1 e P_2 ottenuti aggiungendo in ciascuno un vincolo di forma semplice (ovvero una limitazione sui valori di una singola variabile) che esclude il vertice A (regola di branching). In P_1 si aggiunge il vincolo che una delle variabili a valore frazionario in A sia non superiore alla parte intera di tale valore, mentre in P_2 si aggiunge il vincolo che la stessa variabile sia non inferiore alla parte intera dello stesso valore incrementata di uno. Nel nostro esempio in A sono frazionari i valori di entrambe le variabili. Come regola di scelta si adotta qui quella di prendere la variabile con indice minimo e quindi, nel nostro caso, x_1. Si noti che tale regola non è l'unica possibile e altre più sofisticate possono essere adottate, ad esempio scegliere quella con parte frazionaria più elevata. Qui però non ci preoccuperemo tanto degli aspetti di efficienza dell'algoritmo, che possono essere certamente migliorati con altre regole, ma ci concentreremo sulla descrzione dei principi di base dell'approccio. Per quanto detto, in P_1 si aggiungerà il vincolo $x_1 \leq \lfloor \frac{5}{4} \rfloor = 1$, mentre in P_2 verrà aggiunto il vincolo $x_1 \geq \lfloor \frac{5}{4} \rfloor + 1 = 2$. I due sottoproblemi saranno quindi i seguenti:

$$
\begin{array}{lll}
P_1: \quad \max & x_1 + 3x_2 & \\
& (u_1) & x_1 \geq \dfrac{1}{2} \\
& (u_2) & x_1 + \dfrac{7}{5}x_2 \leq \dfrac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4') & x_1 \leq \left\lfloor \dfrac{5}{4} \right\rfloor = 1 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

$$
\begin{array}{lll}
P_2: \quad \max & x_1 + 3x_2 & \\
& (u_1) & x_1 \geq \dfrac{1}{2} \\
& (u_2) & x_1 + \dfrac{7}{5}x_2 \leq \dfrac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4'') & x_1 \geq \left\lfloor \dfrac{5}{4} \right\rfloor + 1 = 2 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

In Figura 9.4 è rappresentato il rilassamento lineare P_1' di P_1, ottenuto rimuovendo i vincoli di interezza da P_1. In Figura 9.5 è rappresentato il rilassamento lineare P_2' di P_2. Le regioni ammissibili di P_1' (il politopo $CBEF$) e P_2' (il politopo GHD) sono parti (disgiunte) della regione ammissibile di P_0'. La loro unione contiene tutte le soluzioni ammissibili di P_0 ma non coincide con la regione ammissibile di P_0' in quanto manca la zona $1 < x_1 < 2$ nella quale si trova il vertice ottimo A di P_0'.

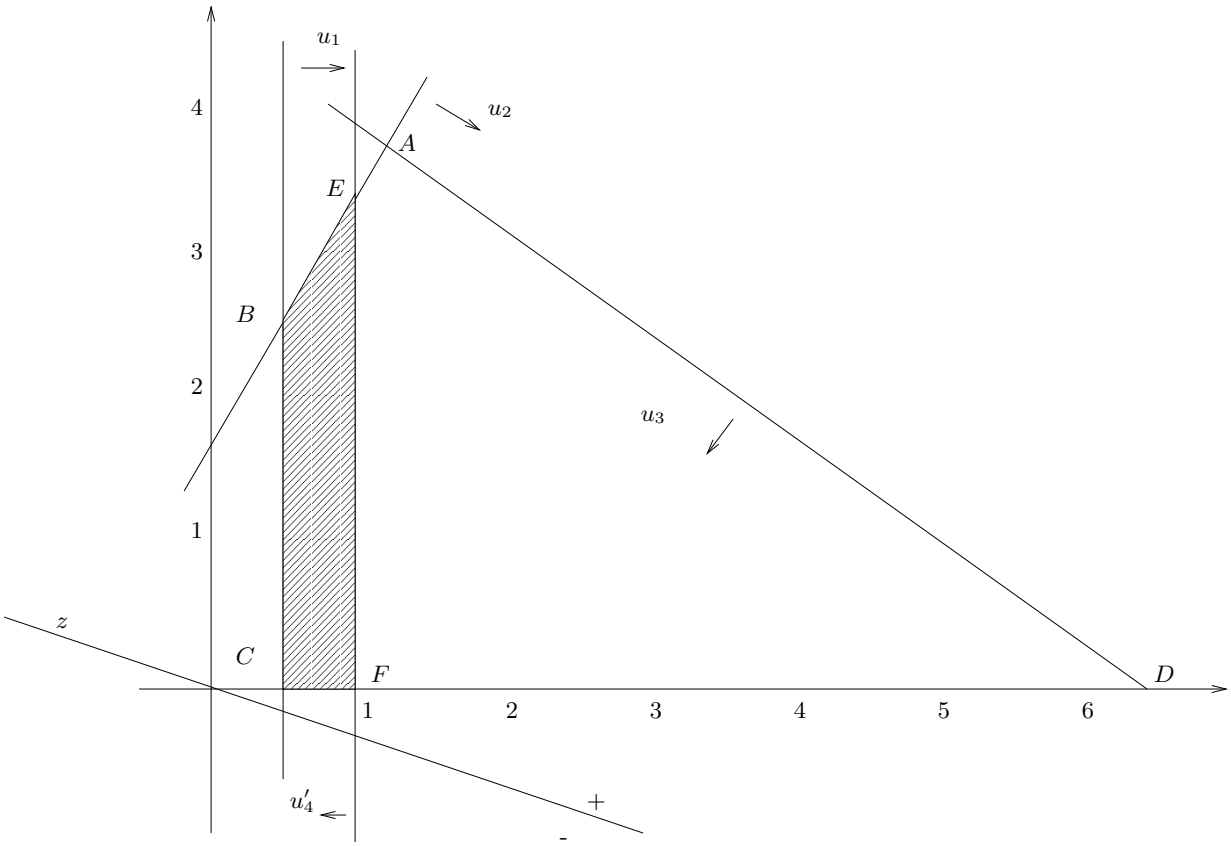


Figura 9.4: Il rilassamento lineare P_1' di P_1, con regione ammissibile $ABEF$.

L'unica soluzione ottima di P_1' è il vertice E $\left(1, \frac{10}{3}\right)$ con valore ottimo pari a $1 + 3\frac{10}{3} = 11$. Si noti che E non ha coordinate intere e quindi non è una soluzione ammissibile per il problema originario P_0. L'unica soluzione ottima del problema P_2' è il vertice H $\left(2, \frac{45}{14}\right)$ con valore ottimo pari a $2 + 3\frac{45}{14} = \frac{163}{14}$. Anche in questo caso H non è soluzione ammissibile del problema P_0.
Possiamo rappresentare quanto fatto sino ad ora attraverso un albero, detto albero di Branch-and-Bound, mostrato in Figura 9.6. I due archi che si diramano dal nodo radice P_0 indicano la suddivisione del problema P_0 nei due

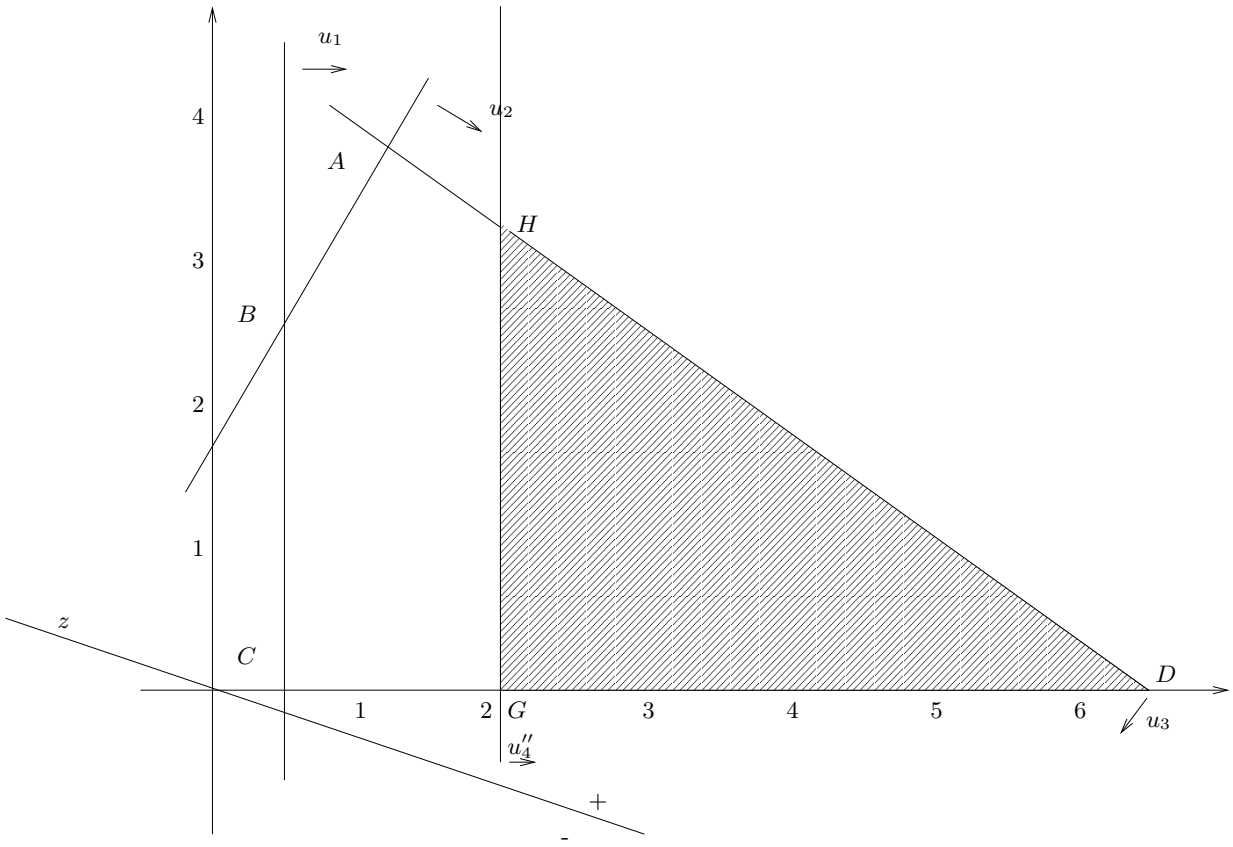


Figura 9.5: Il rilassamento lineare P_2' di P_2, con regione ammissibile GHD.

sottoproblemi P_1 e P_2, ottenuti aggiungendo rispettivamente i vincoli $x_1 \leq 1$ e $x_1 \geq 2$ (riportati lungo gli archi) al problema originario P_0. Al fianco dei due nodi P_1 e P_2 sono indicati i valori $U(P_1)$ e $U(P_2)$ che indicano i valori di *upper bound*, o limitazione superiore, per tali nodi. Per upper bound del nodo P_i $(i = 1, 2)$ si intende che tutte le soluzioni ammissibili del sottoproblema P_i non possono avere valore della funzione obiettivo superiore all'upper bound. Se si riflette su come sono stati ottenuti i valori $U(P_i)$, $(i = 1, 2)$ si capisce che questo è vero. Infatti i valori $U(P_i)$ sono i valori ottimi dei rilassamenti lineari P_i' dei P_i. Quindi tutte le soluzioni ammissibili di P_i' ed in particolare quelle a coordinate intere (ammissibili per P_i) avranno valore della funzione obiettivo che non supera $U(P_i)$. Per il momento non è ancora chiaro che cosa faremo con i valori $U(P_i)$. Questo verrà specificato nel seguito.
Oltre ai valori di upper bound associati a ciascun nodo, si definisce anche un valore unico di *lower bound*, o limitazione inferiore, LB che indica il valore della miglior soluzione ammissibile a coordinate intere tra quelle ottenute come

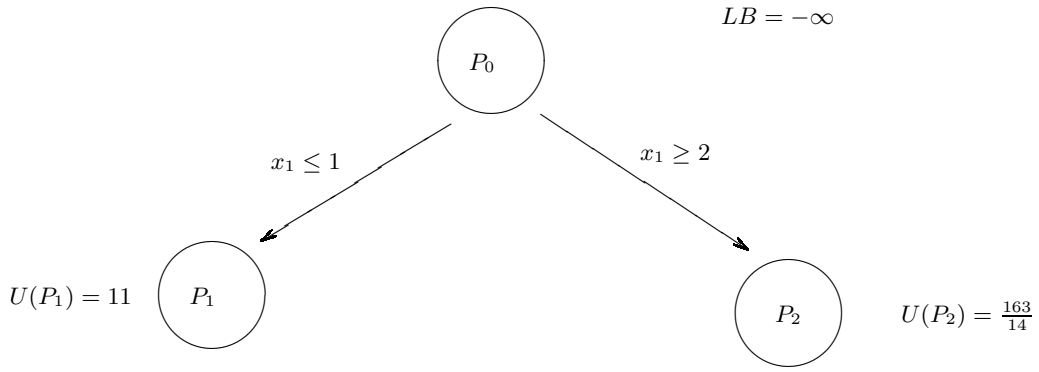


Figura 9.6: L'albero di Branch-and-Bound dopo la prima iterazione dell'algoritmo.

soluzioni ottime dei sottoproblemi. Nel nostro esempio non abbiamo fino ad ora determinato nessuna soluzione ottima di sottoproblemi a coordinate intere (il vertice ottimo E di P_1' ed il vertice ottimo H di P_2' non sono a coordinate intere). In questo caso il valore LB viene posto pari a $-\infty$.

A questo punto possiamo ripetere il ragionamento fatto fino ad ora: dopo aver suddiviso il problema P_0 nei due sottoproblemi P_1 e P_2, possiamo selezionare uno dei due sottoproblemi (o, equivalentemente, uno dei nodi foglia dell'albero) e suddividere questo a sua volta. Per stabilire quale nodo selezionare esistono diverse regole. Qui ne vedremo una soltanto, quella già vista nello schema generale, ovvero selezionare un nodo con upper bound massimo. L'idea su cui si basa questa regola è che ci si aspetta di trovare più facilmente la soluzione ottima del problema in un nodo con un valore di upper bound elevato. Nel nostro esempio il nodo con upper bound più elevato è il nodo P_2. Il sottoproblema P_2 verrà suddiviso in due sottoproblemi P_3 e P_4 ottenuti aggiungendo rispettivamente il vincolo $x_2 \leq \left\lfloor \frac{45}{14} \right\rfloor = 3$ ed il vincolo $x_2 \geq \left\lfloor \frac{45}{14} \right\rfloor + 1 = 4$. Nella determinazione dei vincoli di forma semplice da aggiungere a quelli di P_2 per ottenere P_3 e P_4 la scelta ricade sulla variabile x_2 in quanto è l'unica a coordinate non intere nel

vertice ottimo H di P_2'. Avremo dunque

$$
\begin{array}{lll}
P_3 : \quad \max & x_1 + 3x_2 & \\
& (u_1) & x_1 \geq \dfrac{1}{2} \\
& (u_2) & x_1 + \dfrac{7}{5}x_2 \leq \dfrac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4'') & x_1 \geq 2 \\
& (u_5') & x_2 \leq \left\lfloor \dfrac{45}{14} \right\rfloor = 3 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

$$
\begin{array}{lll}
P_4 : \quad \max & x_1 + 3x_2 & \\
& (u_1) & x_1 \geq \dfrac{1}{2} \\
& (u_2) & x_1 + \dfrac{7}{5}x_2 \leq \dfrac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4'') & x_1 \geq 2 \\
& (u_5'') & x_2 \geq \left\lfloor \dfrac{45}{14} \right\rfloor + 1 = 4 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

Le regioni ammissibili di P_3' (il politopo $GKJD$) e P_4' (l'insieme vuoto) sono illustrate nelle Figure 9.7 e 9.8. Come in precedenza l'unione delle regioni ammissibili di P_3' e P_4' contiene tutte le soluzioni ammissibili di P_2 ma esclude tutte le soluzioni ammissibili di P_2' nella zona $3 < x_2 < 4$, dove si trova il vertice ottimo H di P_2'.
L'ottimo di P_3' si trova nel punto J $\left(\frac{23}{10}, 3\right)$ con valore ottimo $\frac{113}{10}$. Per quel che riguarda P_4' si vede subito che la regione ammissibile è vuota ed in tal caso il nodo viene cancellato e non verrà ulteriormente esplorato. Infatti esso, non contenendo soluzioni ammissibili per P_4', non può certamente contenere soluzioni ammissibili per P_4.
Il nostro albero si presenterà ora nella forma di Figura 9.9. Anche la soluzione ottima J di P_3' non è a coordinate intere e quindi il valore LB continuerà ad essere pari a $-\infty$.

A questo punto si ripete nuovamente il ragionamento suddividendo un sottoproblema relativo ad uno dei nodi foglia dell'albero in Figura 9.9 non ancora cancellato. I nodi foglia sono 3 (P_1, P_3 e P_4) ma P_4 è già stato cancellato. La scelta quindi si restringe a P_1 e P_3. Come in precedenza, si seleziona quello con

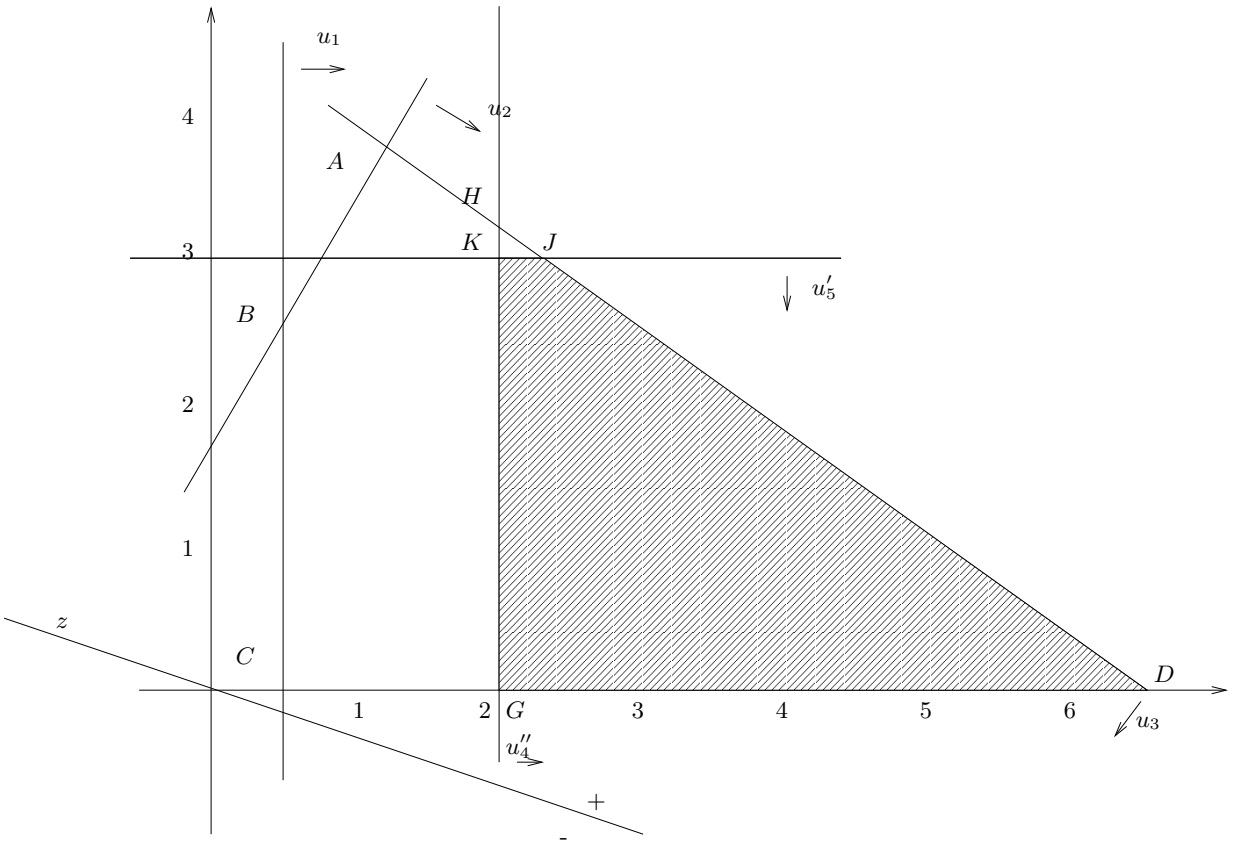


Figura 9.7: Il rilassamento lineare P'_3 di P_3, con regione ammissibile $GKJD$.

upper bound maggiore e quindi il nodo P_3. Il sottoproblema P_3 viene suddiviso nei due sottoproblemi P_5 e P_6 ottenuti aggiungendo ai vincoli di P_3 rispettivamente il vincolo $x_1 \leq \lfloor \frac{23}{10} \rfloor = 2$ ed il vincolo $x_1 \geq \lfloor \frac{23}{10} \rfloor + 1 = 3$. La scelta della variabile x_1 nella definizione di questi vincoli semplici è l'unica possibile,

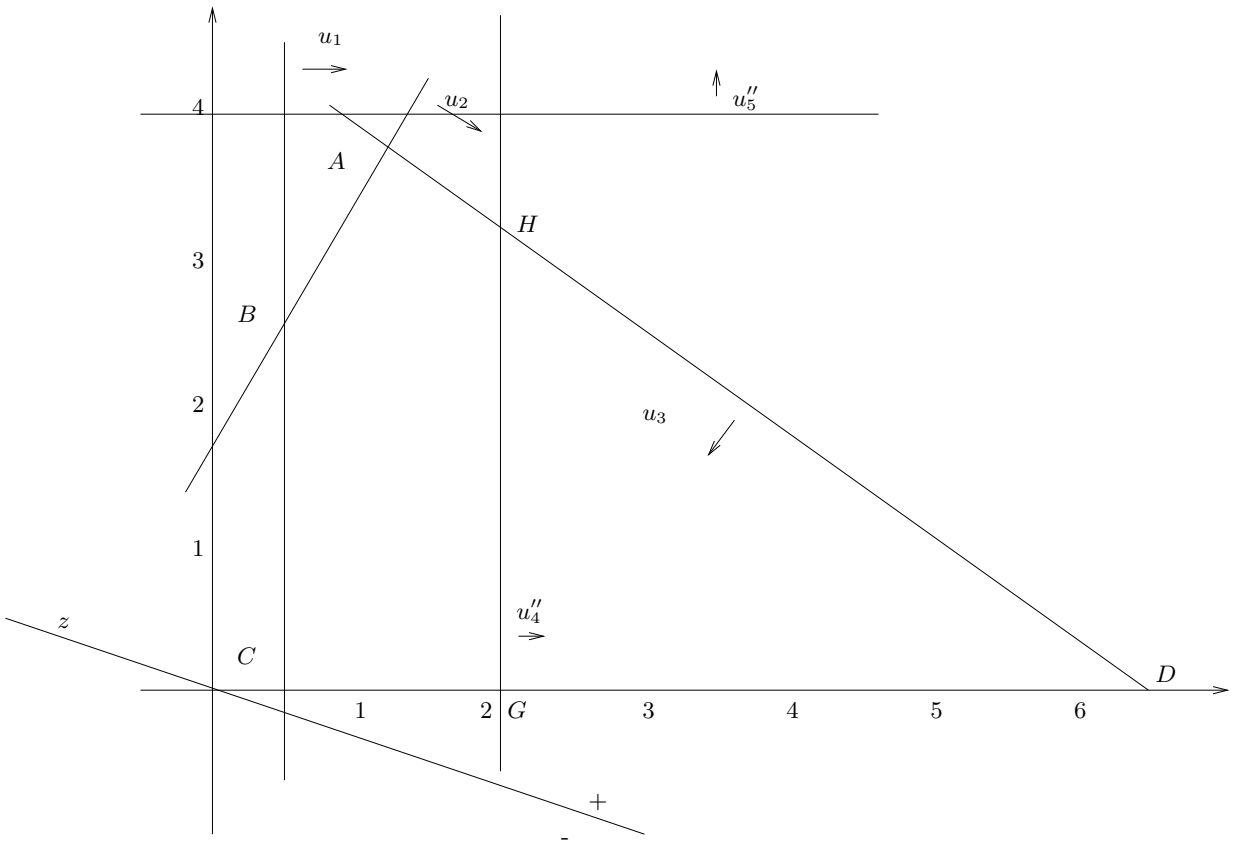


Figura 9.8: Il rilassamento lineare P_4' di P_4, con regione ammissibile vuota.

essendo x_1 la sola variabile frazionaria nella soluzione ottima J di P_3'.

$$
\begin{array}{lll}
P_5 \ : \quad \max & x_1 + 3x_2 & \\
& (u_1) & x_1 \geq \dfrac{1}{2} \\
& (u_2) & x_1 + \dfrac{7}{5}x_2 \leq \dfrac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4'') & x_1 \geq 2 \\
& (u_5') & x_2 \leq 3 \\
& (u_6') & x_1 \leq \left\lfloor \dfrac{23}{10} \right\rfloor = 2 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

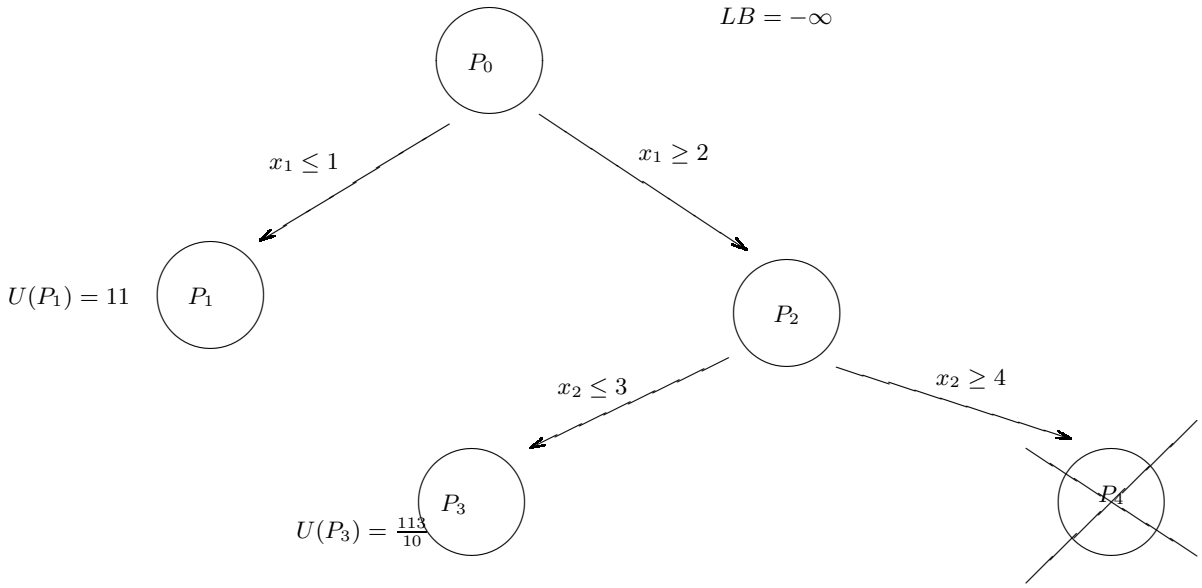


Figura 9.9: L'albero di Branch-and-Bound dopo la seconda iterazione dell'algoritmo.

$$
\begin{array}{lll}
P_6: & \max & x_1 + 3x_2 \\
& (u_1) & x_1 \geq \frac{1}{2} \\
& (u_2) & x_1 + \frac{7}{5}x_2 \leq \frac{13}{2} \\
& (u_3) & -5x_1 + 3x_2 \leq 5 \\
& (u_4'') & x_1 \geq 2 \\
& (u_5') & x_2 \leq 3 \\
& (u_6'') & x_1 \geq \left\lfloor \frac{23}{10} \right\rfloor + 1 = 3 \\
& & x_1, x_2 \geq 0 \\
& & x_1, x_2 \in I
\end{array}
$$

I rilassamenti lineari P_5' (con regione ammissibile il segnento KG) e P_6' (con regione ammissibile il politopo LMD) di P_5 e P_6 sono illustrati nelle Figure 9.10 e 9.11. Il problema P_5' ha soluzione ottima nel vertice K $(2, 3)$ con valore ottimo $2 + 3 * 3 = 11$. Il problema P_6' ha soluzione ottima L $\left(3, \frac{5}{2}\right)$ con valore ottimo $3 + 3\frac{5}{2} = \frac{21}{2}$.
L'albero di Branch-and-Bound si presenta ora nella forma di Figura 9.12. Notiamo che la soluzione K di P_5' è a coordinate intere ed è dunque una soluzione ammissibile per P_0. Questo ci permette di aggiornare il valore LB che sarà ora

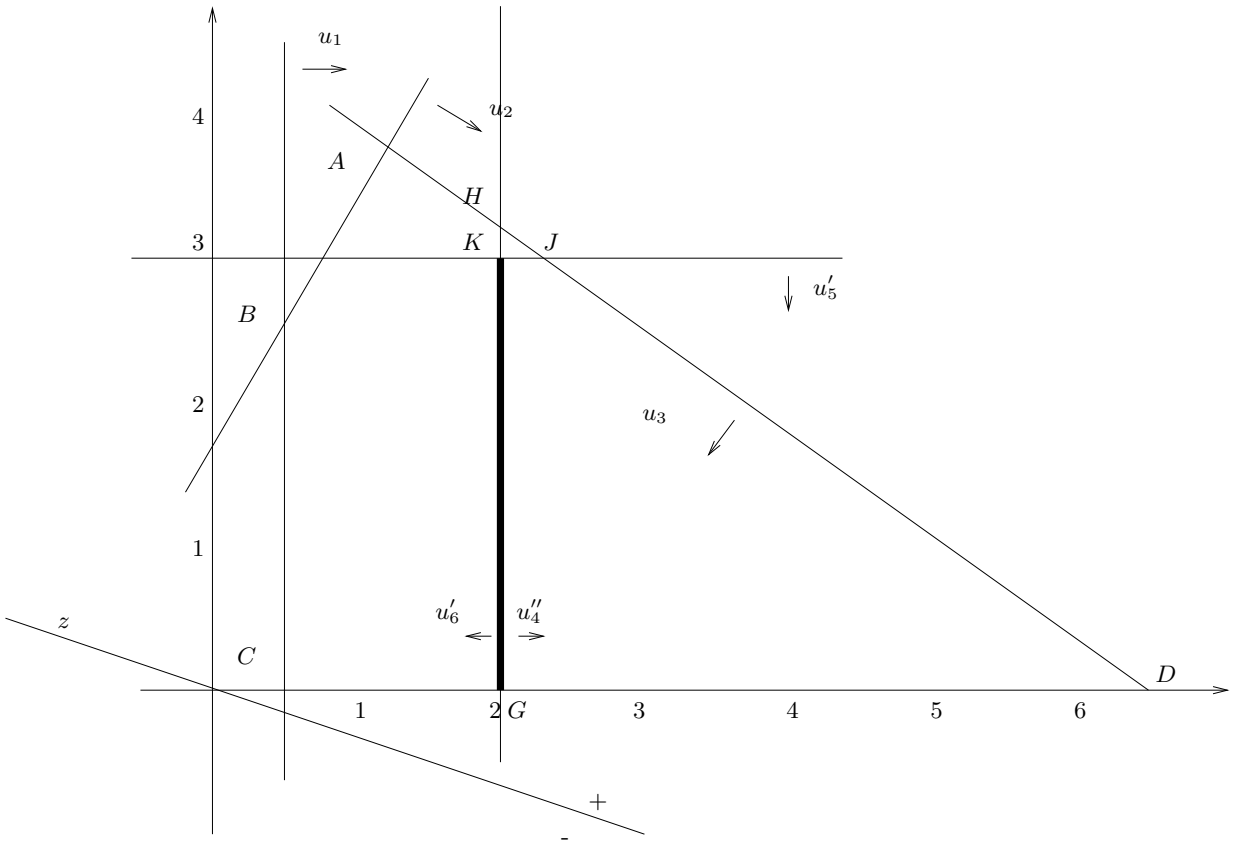


Figura 9.10: Il rilassamento lineare P_5' di P_5, con regione ammissibile KG.

pari al valore della funzione obiettivo nel punto K, ovvero pari a 11.
A questo punto osserviamo che il nodo P_5 può essere cancellato. Infatti il punto K, a coordinate intere, è soluzione ottima di P_5' e quindi è anche soluzione ottima di P_5 e non possiamo sperare di trovare, nella regione ammissibile di P_5, soluzioni ammissibili di P_0 con un valore della funzione obiettivo più elevato rispetto al valore in K.
Ma anche gli altri nodi foglia non ancora cancellati (P_1 e P_6) possono essere cancellati. Consideriamo, ad esempio, P_1. L'upper bound per le soluzioni ammissibili di P_1 è il valore 11. Non possiamo quindi sperare di trovare in P_1 delle soluzioni ammissibili di P_0 che superino il valore $LB = 11$. Quindi possiamo evitare di esplorare ulteriormente il nodo P_1 e possiamo cancellare tale nodo. Analogo discorso vale per il nodo P_6 dove si ha che $U(P_6) = \frac{21}{2} < 11 = LB$. Quindi anche questo nodo può essere cancellato. Aggiornando l'albero con le nuove cancellazioni, si ottiene l'albero in Figura 9.13. Notando che tutti i nodi foglia sono stati cancellati, ci possiamo arrestare e restituire come soluzione

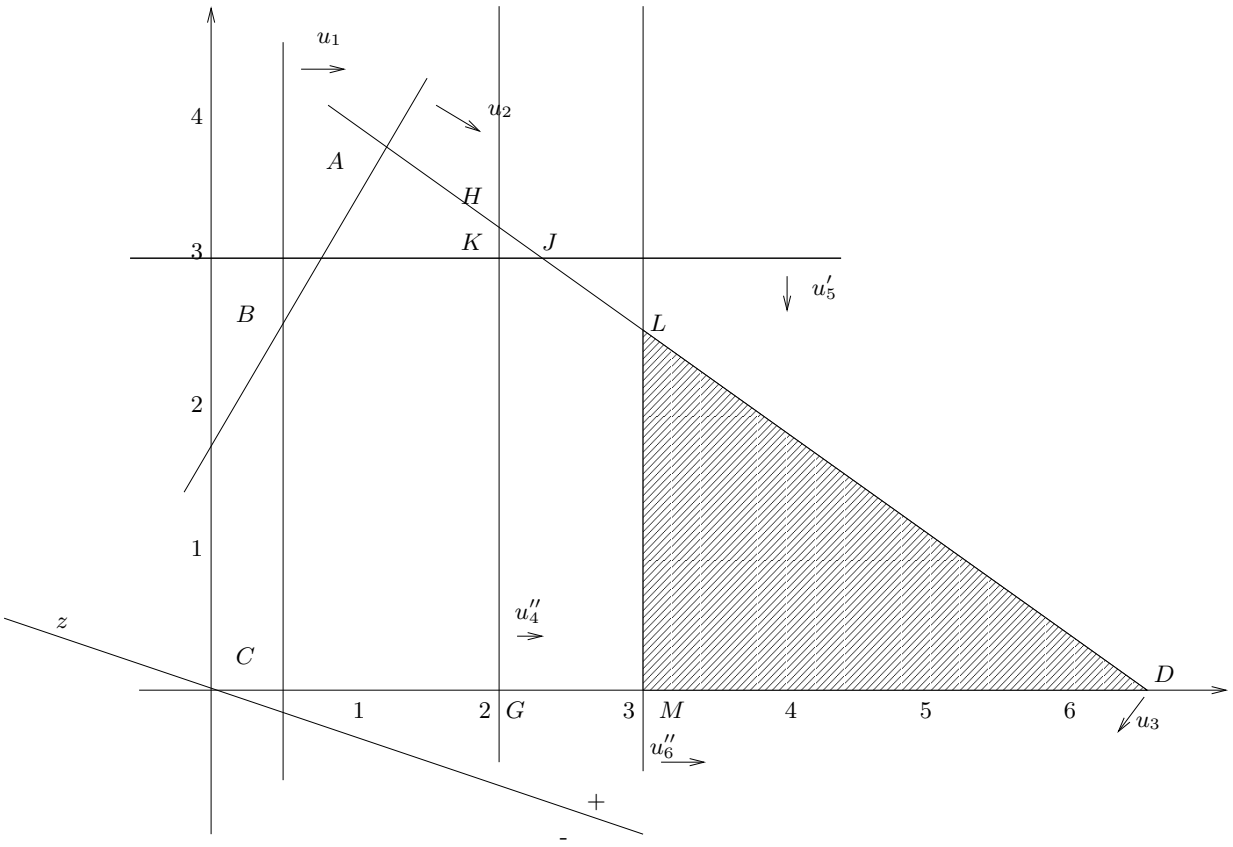


Figura 9.11: Il rilassamento lineare P_6' di P_6, con regione ammissibile LMD.

ottima del problema la miglior soluzione ammissibile trovata, ovvero il punto K $(2, 3)$ con valore ottimo pari a 11.
Come ultima osservazione si noti che il nodo P_1 ha upper bound esattamente pari a LB. Quindi, pur non potendo sperare di trovare in P_1 una soluzione ammissibile di P_0 che sia *strettamente superiore* a $LB = 11$, potremmo comunque sperare di trovarne una con valore *uguale* a 11. In realtà abbiamo già osservato che il rilassamento lineare P_1' di P_1 ha come unica soluzione ottima il vertice, a coordinate non intere, E e quindi tutte le altre soluzioni ammissibili di P_1', comprese quelle a coordinate intere ammissibili per P_1, hanno valore della funzione obiettivo strettamente minore di 11.

9.3.2 Cancellazione dei nodi

Prima di formalizzare l'algoritmo di Branch-and-Bound rivediamo attraverso l'esempio considerato i modi con cui è possibile cancellare i nodi foglia in un

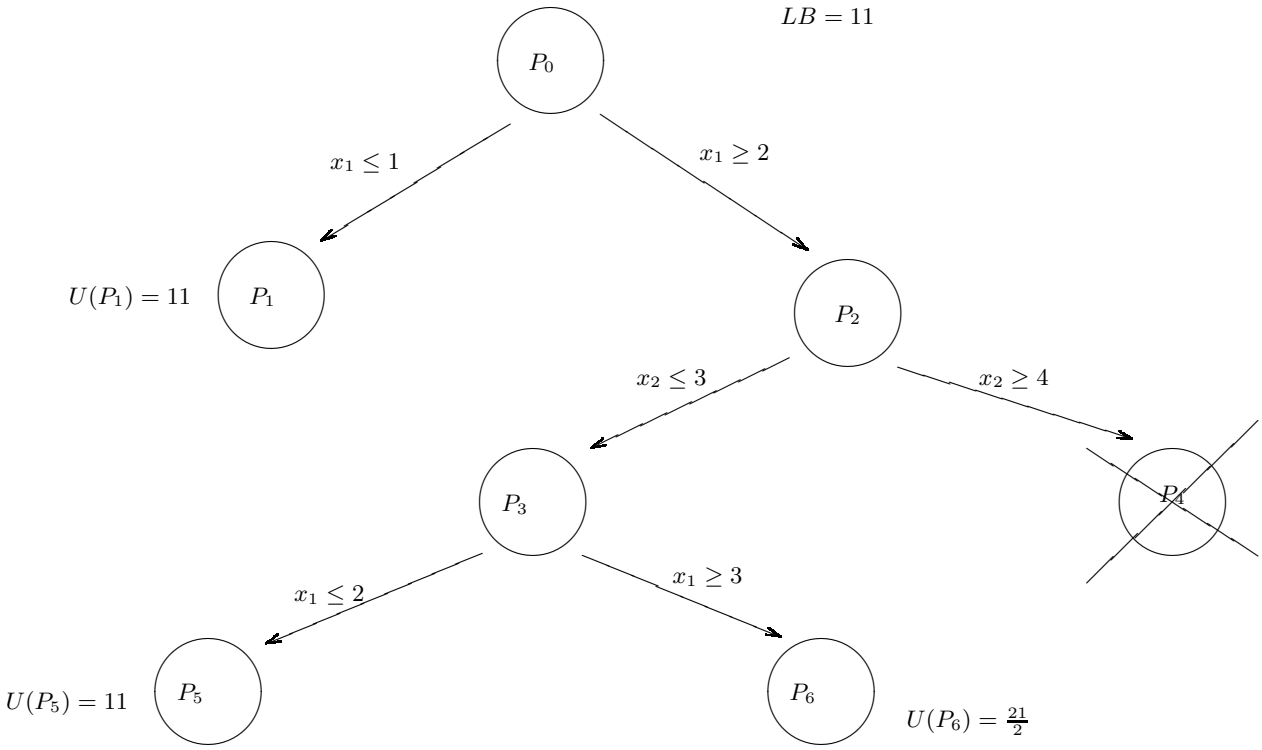


Figura 9.12: L'albero di Branch-and-Bound dopo la terza iterazione dell'algoritmo.

albero di Branch-and-Bound. Dato un nodo P, esso può essere cancellato se almeno una delle seguenti condizioni è soddisfatta.

1. Il corrispondente rilassamento lineare P' ha regione ammissibile vuota e quindi nel nodo P non vi possono essere soluzioni ammissibili per il problema di PLI originario, cioè il nodo radice P_0. Nel nostro esempio questo si verifica nel nodo P_4.

2. Il rilassamento lineare P' di P ha soluzione ottima intera. Tale soluzione è la migliore soluzione ammissibile del problema di PLI originario P_0 che sia possibile trovare nel nodo P (vedi il nodo P_5 nell'esempio).Questa soluzione può anche essere usata per aggiornare, eventualmente, il lower bound LB.

3. Si ha che $U(P) \leq LB$. In tal caso l'upper bound $U(P)$ nel nodo P, ovvero il valore ottimo del rilassamento lineare P' di P, non supera il lower bound LB (il valore della miglior soluzione ammissibile trovata dall'algoritmo) e quindi nessuna soluzione ammissibile del problema P_0 che sia contenuta

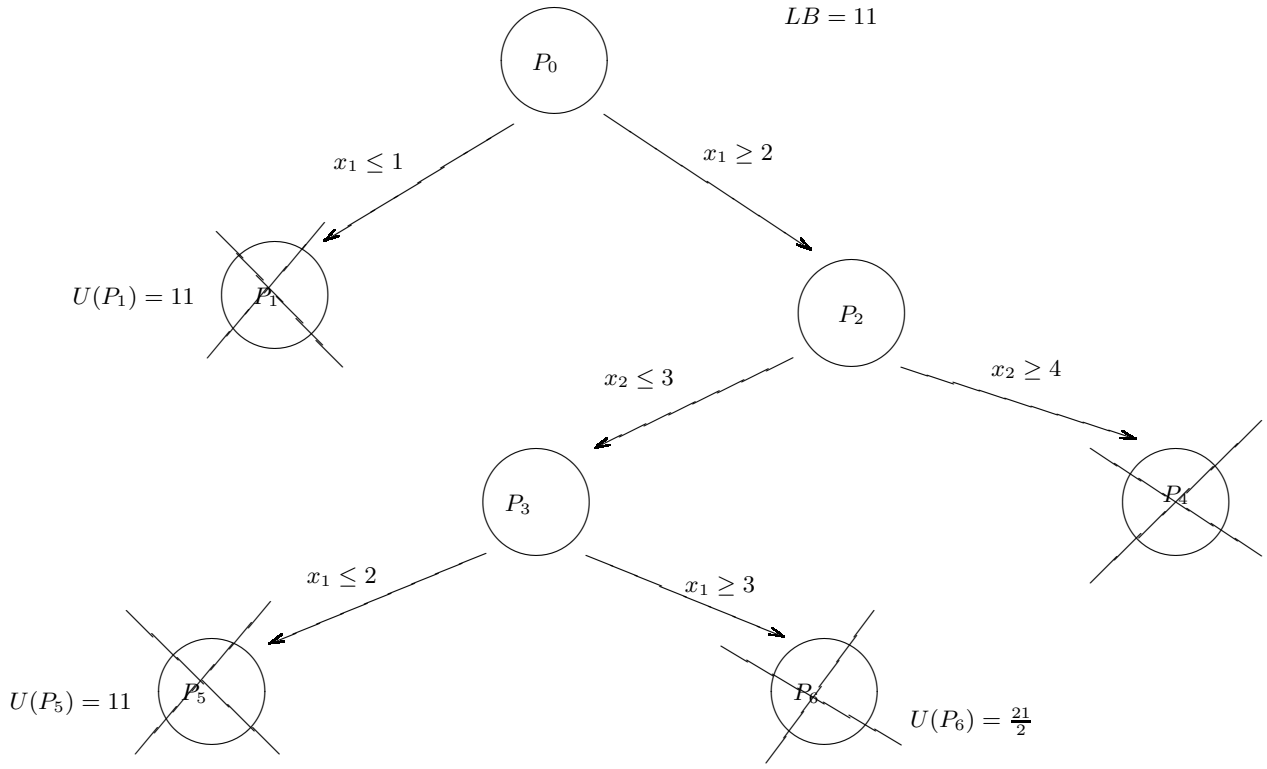


Figura 9.13: L'albero di Branch-and-Bound al momento in cui l'algoritmo viene fermato.

in P può superare il valore LB. Se, in particolare, si ha $U(P) < LB$ si ha che nessuna soluzione ammissibile del problema P_0 che sia contenuta in P non solo non può superare il valore LB ma non lo può neppure eguagliare. Il caso $U(P) = LB$ si verifica nel nostro esempio nel nodo P_1, mentre il caso $U(P) < LB$ si verifica nel nodo P_6.

9.3.3 Branch-and-Bound: l'algoritmo generale

Siamo ora pronti a generalizzare quanto visto nell'esempio e a formalizzare l'algoritmo Branch-and-Bound. Il generico problema di PLI P_0 si presenta nella seguente forma

$$\begin{array}{lll} P_0 \ : \quad \max & \sum_{i=1}^{n} c_i x_i & \\ & \sum_{i=1}^{n} a_{ij} x_i \leq b_j & j = 1, \ldots, m \\ & x_i \geq 0 & i = 1, \ldots, n \\ & x_i \in Z & i = 1, \ldots, n \end{array}$$

L'algoritmo di Branch-and-Bound è il seguente.

Inizializzazione Si inizializzi l'insieme $\mathcal{F}$ dei nodi foglia con il problema originario P_0, cioè si ponga $\mathcal{F} = \{P_0\}$. Si risolva il rilassamento lineare P_0' di P_0 e sia $(x_1^*(P_0), \ldots, x_n^*(P_0))$ la soluzione ottima trovata e $U(P_0)$ il corrispondente valore ottimo.
Se tutti i valori $x_i^*(P_0),\ i = 1, \ldots, n$ sono interi, STOP: la soluzione $(x_1^*(P_0), \ldots, x_n^*(P_0))$ è soluzione ottima anche per P_0 ed il valore ottimo di P_0 è pari a $U(P_0)$.
Altrimenti si ponga $LB = -\infty$ e si vada al Passo 1.

Passo 1 Si selezioni nell'insieme $\mathcal{F}$ un nodo $Q \in \mathcal{F}$. Anche se esistono altre regole per effettuare la selezione, qui adotteremo quella di scegliere il nodo $Q \in \mathcal{F}$ con valore dell'upper bound massimo, ovvero

$$U(Q) = \max_{P \in \mathcal{F}} U(P)$$

e sia $(x_1^*(Q), \ldots, x_n^*(Q))$ una soluzione ottima del rilassamento lineare Q' di Q.

Passo 2 Si selezioni una variabile $x_i^*(Q)$ a valore frazionario. Nel caso vi siano più variabili $x_i^*(Q)$ con valore frazionario, esistono diverse regole per effettuare la scelta. Qui, per semplicità, si adotterà quella di scegliere la variabile con indice i minimo.

Passo 3 Si rimuova il nodo Q da $\mathcal{F}$ (cioè $\mathcal{F} = \mathcal{F} \setminus \{Q\}$) e lo si suddivida in due nuovi nodi Q_1 e Q_2 ottenuti aggiungendo ai vincoli di Q rispettivamente il vincolo $x_i \leq \lfloor x_i^*(Q) \rfloor$ (in Q_1) ed il vincolo $x_i \geq \lfloor x_i^*(Q) \rfloor + 1$ (in Q_2).

Passo 4 Per ciascuno dei due nuovi nodi $Q_i,\ i = 1, 2$ si risolva il rilassamento lineare Q_i'.
Se Q_i' ha regione ammissibile vuota, si cancella il nodo Q_i e non lo si aggiunge a $\mathcal{F}$, altrimenti lo si aggiunge a $\mathcal{F}$.
Altrimenti se la soluzione ottima $(x_1^*(Q_i), \ldots, x_n^*(Q_i))$ è a coordinate intere si cancelli il nodo Q_i senza aggiungerlo a $\mathcal{F}$ ed inoltre, se $U(Q_i) > LB$ si ponga

$$LB = U(Q_i) \quad \text{e} \quad y_1 = x_1^*(Q_i), \ldots, y_n = x_n^*(Q_i).$$

Altrimenti, se la soluzione ottima di Q_i' non è a coordinate intere, si aggiunga Q_i a $\mathcal{F}$, ovvero si ponga $\mathcal{F} = \mathcal{F} \cup \{Q_i\}$.

Passo 5 Si cancellino tutti i nodi in $\mathcal{F}$ con valore dell'upper bound non superiore a LB, ovvero si ponga

$$\mathcal{F} = \mathcal{F} \setminus \{P \in \mathcal{F} \ : \ U(P) \leq LB\}.$$

Passo 6 Se $\mathcal{F} = \emptyset$, STOP: se LB ha valore pari a $-\infty$, allora il problema P_0 non ha soluzioni ammissibili, altrimenti il valore LB è il valore ottimo del problema P_0 e $(y_1, \ldots, y_n)$ è una soluzione ottima di tale problema.
Altrimenti si ritorni al Passo 1.

Capitolo 10

Problemi di ottimizzazione e complessità

Parlando degli algoritmi branch-and-bound abbiamo già discusso di problemi di ottimizzazione. Vediamo di darne una definizione formale.

Definizione 3 *Un problema di ottimizzazione è formato da un insieme di istanze, dove ogni istanza è rappresentata da una coppia* (f, S) *con*

$$f : S \to R$$

detta funzione obiettivo *e* S *detta* regione ammissibile*. Se il problema è di massimo, allora l'istanza si rappresenta nel modo seguente*

$$\max_{x \in S} f(x)$$

e risolvere l'istanza vuol dire trovare un $x^* \in S$ *tale che*

$$f(x^*) \geq f(x) \quad \forall\ x \in S;$$

se il problema è di minimo, allora l'istanza si rappresenta nel modo seguente

$$\min_{x \in S} f(x)$$

e risolvere l'istanza vuol dire trovare un $x^* \in S$ *tale che*

$$f(x^*) \leq f(x) \quad \forall\ x \in S.$$

In entrambi i casi il punto x^* *viene chiamato* ottimo (globale) *dell'istanza, mentre* $f(x^*)$ *viene detto* valore ottimo *dell'istanza.*

I problemi di ottimizzazione possono essere raggruppati in due grandi categorie.

Ottimizzazione Combinatoria In ogni istanza la regione ammissibile S contiene un numero finito o un'infinità numerabile di punti.

Ottimizzazione Continua La regione ammissibile S può contenere un'infinità non numerabile di punti.

Di seguito forniamo un elenco esteso di problemi di ottimizzazione, alcuni dei quali ci sono già noti.

Problemi di PL e PLI

Abbiamo già incontrato i problemi di PL e PLI. Le istanze dei problemi di PL (Programmazione Lineare) in forma canonica (alla quale però sappiamo che possiamo sempre ricondurci) hanno la seguente forma: dati $A \in R^{m \times n}$, $b \in R^m$, $c \in R^n$

$$\begin{aligned} &\max\ cx \\ &Ax \leq b \\ &x \geq 0 \end{aligned}$$

dove

$$f(x) = cx \quad S = \{x \in R^n\ :\ Ax \leq b,\ x \geq 0\}.$$

In pratica, tutte le istanze dei problemi di PL possono essere ottenute variando il numero m di vincoli, il numero n di variabili e i dati contenuti nella matrice A e nei vettori b e c.

Le istanze dei problemi di PLI (Programmazione Lineare Intera) sono simili a quelle di PL, l'unica differenza è l'imposizione tra i vincoli dell'interezza delle variabili:

$$\begin{aligned} &\max\ cx \\ &Ax \leq b \\ &x \geq 0 \qquad x \in Z^n \end{aligned}$$

Mentre i problemi di PLI sono di ottimizzazione combinatoria, i problemi di PL sono di ottimizzazione continua.

`SHORTEST PATH`: problema del cammino a costo minimo

Nei problemi `SHORTEST PATH`, dato un grafo orientato $G = (V, A)$ con costo (distanza) d_{ij} per ogni $(i, j) \in A$ e dati due nodi $s, t \in V$, $s \neq t$, vogliamo individuare un cammino orientato da s a t di costo minimo.

`MST`: problema dell'albero di supporto a peso minimo

Dato un grafo non orientato $G = (V, E)$ con pesi w_{ij} per ogni $(i, j) \in E$, si vuole determinare tra tutti gli alberi di supporto $T = (V, E_T)$ del grafo quello con peso complessivo $\sum_{(i,j) \in E_T} w_{ij}$ minimo.

`CLIQUE`

Dato un grafo $G = (V, E)$, vogliamo individuare un sottinsieme $C \subseteq V$ dei nodi tali che

$$\forall\ i, j \in C,\ i \neq j\ :\ (i, j) \in E$$

(il sottografo indotto da C è completo) di cardinalità $\mid C \mid$ massima.

`TSP`: il problema del commesso viaggiatore

Abbiamo già discusso tale problema con il relativo modello nel Capitolo 2. Dato un grafo orientato completo $G = (V, A)$ con le relative distanze degli archi rappresentate dagli interi non negativi d_{ij}, per ogni $(i, j) \in A$, individuare nel grafo un circuito hamiltoniano (ciclo che tocca tutti i nodi del grafo una e una sola volta) di distanza totale (somma delle distanze dei suoi archi) minima. In pratica, si tratta di scegliere una permutazione $i_1, \ldots, i_n$ degli $n = \mid V \mid$ nodi del grafo tale che il circuito

$$i_1 \rightarrow \cdots \rightarrow i_n \rightarrow i_1$$

sia di lunghezza minima. Si noti che, fissando il nodo iniziale i_1, il numero di possibili circuiti hamiltoniani è pari a $(n-1)!$. Tra le sottoclassi di questo problema citiamo il `TSP` *simmetrico* ($d_{ij} = d_{ji}$ per ogni i, j), che a sua volta contiene la sottoclasse del `TSP` *metrico* dove in più le distanze soddisfano la diseguaglianza triangolare

$$d_{ij} \leq d_{ik} + d_{kj} \quad \forall\ i, j, k \text{ distinti.}$$

`KNAPSACK`: il problema dello zaino

Dati n oggetti aventi come pesi gli interi positivi $p_1, \ldots, p_n$ e come valori gli n interi positivi $v_1, \ldots, v_n$ e dato uno zaino con capacità pari a un intero positivo b, vogliamo individuare un sottinsieme K degli n oggetti con peso complessivo $\sum_{i \in K} p_i$ non superiore a b e valore complessivo $\sum_{i \in K} v_i$ massimo.

10.1 Difficoltà dei problemi di ottimizzazione

Dato un problema di ottimizzazione il nostro scopo è ovviamente risolverlo, restituendone una soluzione ottima e il valore ottimo. Per risolverlo abbiamo bisogno di un *algoritmo* di risoluzione $\mathcal{A}$ per il problema, ovvero una procedura che, ricevuti in input i dati che definiscono un'istanza, restituisce in output proprio una soluzione ottima e il valore ottimo di quell'istanza.

Se ci limitiamo ai problemi di ottimizzazione combinatoria in cui la regione ammissibile è costituita da un numero finito di elementi (vale per i problemi `MST`, `SHORTEST PATH`, `CLIQUE`, `TSP`, `KNAPSACK`), è abbastanza semplice identificare un algoritmo di risoluzione. Infatti se la regione ammissibile S contiene un

numero finito di elementi la seguente procedura, detta di *enumerazione completa*, risolve il problema: valutare la funzione f per ogni elemento in S e restituire l'elemento con valore di f minimo (o massimo se il problema è di massimo). Tuttavia una semplice osservazione metterà in luce i limiti di questo approccio. Consideriamo il problema `TSP` su un grafo con numero di nodi $n = 22$. Abbiamo osservato come in tale grafo il numero di circuiti hamiltoniani è pari a $(n-1)! = 21! > 10^{19}$. Supponiamo (ottimisticamente) che la valutazione della funzione obiettivo per un singolo circuito hamiltoniano richieda un nanosecondo (10^{-9} secondi). Ne consegue che la valutazione di tutti i circuiti hamiltoniani richiede più di 10^{10} secondi che corrispondono ad un tempo superiore ai 200 anni. Ancora più impressionante è notare quello che succede se si aumenta di uno il numero di nodi: il tempo richiesto supera i 4000 anni! Ciò dimostra che sebbene sia sempre possibile teoricamente risolvere tali problemi, in pratica dobbiamo fare i conti con dei limiti temporali e quindi, da un punto di vista pratico, si parlerà di problema risolvibile con una data procedura solo nel caso in cui la procedura restituisca una soluzione in tempi ragionevoli. Ciò rende la procedura di enumerazione completa accettabile solo per istanze di problemi di dimensioni limitate (quindi, ad esempio, per grafi con pochi nodi per i problemi `TSP`). La domanda successiva è quindi la seguente: esistono altre procedure che consentono di risolvere in tempi ragionevoli istanze dei problemi con dimensioni molto più elevate? La risposta è: dipende dal problema. È in generale vero che esistono procedure molto più efficienti dell'enumerazione completa ma mentre per alcuni (come il problema `MST`) si possono risolvere in tempi ragionevoli istanze di grandi dimensioni, per altri, come il problema `KNAPSACK` e, ancor più, per il problema `TSP` *può* essere difficile risolvere anche istanze di dimensioni non troppo elevate (si osservi il *può*: con algoritmi sofisticati si sono risolte anche istanze di `TSP` con più di 15000 nodi, ma gli stessi algoritmi possono essere messi in crisi da istanze più piccole). Tutto ciò ha una formulazione ben precisa nella teoria della complessità, alla quale giungeremo dopo aver discusso di complessità degli algoritmi.

10.1.1 Complessità degli algoritmi di risoluzione

Ad ogni istanza I di un problema di ottimizzazione combinatoria è associata una dimensione $dim(I)$ che corrisponde alla quantità di memoria necessaria per memorizzare (in codifica binaria) tale istanza. Per esempio, l'istanza di un problema `KNAPSACK` ha una dimensione pari alla quantità di memoria necessaria per memorizzare in codice binario i pesi e i valori degli oggetti e la capacità dello zaino.

Consideriamo ora una procedura di risoluzione o algoritmo $\mathcal{A}$ per il problema (si pensi, ad esempio, alla già citata procedura di enumerazione completa). La risoluzione dell'istanza I con l'algoritmo $\mathcal{A}$ richiederà un certo numero di operazioni (e quindi un certo tempo) indicato con $numop_{\mathcal{A}}(I)$. Fissata una dimensione k vi saranno diverse istanze di dimensione k. L'analisi *worst case* definisce il tempo $t_{\mathcal{A}}(k)$ necessario all'algoritmo $\mathcal{A}$ per risolvere istanze di dimen-

$g(k)$	$k=10$	$k=20$	$k=30$	$k=40$	$k=50$
$\log_2(k)$	3.32	4.32	4.90	5.32	5.64
k	10	20	30	40	50
k^2	100	400	900	1600	2500
k^3	1000	8000	27000	64000	125000
2^k	1024	$> 10^6$	$> 10^9$	$> 10^{12}$	$> 10^{15}$
$k!$	3628800	$> 10^{18}$	$> 10^{32}$	$> 10^{47}$	$> 10^{64}$

Tabella 10.1: Crescita di alcune funzioni g al crescere di k

sione k come il massimo tra tutti i tempi di esecuzione di istanze di dimensione k, cioè

$$t_{\mathcal{A}}(k) = \max_{I:\ dim(I)=k} numop_{\mathcal{A}}(I).$$

Tipicamente non si conosce il valore esatto della funzione $t_{\mathcal{A}}(k)$ ma se ne conosce l'ordine di grandezza. Si dice che la funzione $t_{\mathcal{A}}(k) = O(g(k))$, ovvero che $t_{\mathcal{A}}(k)$ è dell'ordine di grandezza della funzione $g(k)$, se esiste una costante $u > 0$ tale che

$$t_{\mathcal{A}}(k) \leq ug(k).$$

Consideriamo ora la Tabella 10.1.1 con diverse possibili funzioni g e diversi valori di k. Come si vede dalla tabella, mentre fino a $g(k) = k^3$ la crescita di g al crescere di k è ragionevole, per $g(k) = 2^k$ e ancor più per $g(k) = k!$, la crescita è rapidissima. Un algoritmo per il quale $t_{\mathcal{A}}(k)$ fosse dell'ordine di grandezza di 2^k oppure $k!$ si dice che ha *complessità esponenziale*. È evidente che un tale algoritmo consente di risolvere in tempi ragionevoli solo istanze di dimensioni limitate. Per questa ragione si cercano per i problemi algoritmi di risoluzione con *complessità polinomiale*, in cui cioè la funzione $t_{\mathcal{A}}$ è dell'ordine di grandezza di un polinomio k^p per un qualche esponente p. Ovviamente, tanto maggiore è l'esponente p del polinomio, quanto più rapidamente cresce il numero di operazioni dell'algoritmo al crescere di k e quindi quanto più piccole sono le dimensioni dei problemi che l'algoritmo è in grado di risolvere in tempi ragionevoli (già per $p = 4$ la crescita è piuttosto rapida). Tuttavia la crescita polinomiale è sempre preferibile a una esponenziale. A questo punto ci possiamo chiedere se, dato un problema di ottimizzazione, possiamo sempre trovare un algoritmo con complessità polinomiale che lo risolva. La risposta è: forse, ma è molto improbabile. Vedremo di chiarire nel seguito il senso di questa risposta introducendo qualche elemento di *teoria della complessità*. Prima però prendiamo in esame algoritmi di risoluzione per alcuni dei problemi di ottimizzazione precedentemente discussi.

10.1.2 Algoritmi per `SHORTEST PATH`

Per questo problema presenteremo due algoritmi:

- Algoritmo di Dijkstra: valido solo se $d_{ij} \geq 0 \ \forall \ (i,j) \in A$. Restituisce i cammini minimi tra un nodo fissato $s \in V$ e tuuti gli altri nodi del grafo.

- Algoritmo di Floyd-Warshall: valido anche per distanze negative *a patto che non ci siano cicli di lunghezza negativa*. Restituisce i cammini minimi tra tutte le coppie di nodi del grafo *se il grafo non contiene cicli a costo negativo*. In quest'ultimo caso restituisce un ciclo a costo negativo.

Nel seguito si supporrà sempre che $d_{ij} = +\infty$ per ogni $(i,j) \notin A$ e indicheremo con n la cardinalità di V.

Algoritmo di Dijkstra

Inizializzazione Poni

$$W = \{s\}, \quad \rho(s) = 0, \quad e(s) = -$$

e per ogni $y \in V \setminus \{s\}$

$$\rho(y) = d_{sy} \quad e(y) = s.$$

1 Se $W = V$, allora STOP, altrimenti vai al Passo 2.

2 Sia

$$x \in \arg\min\{\rho(y) \ : \ y \notin W\}.$$

3 Poni $W = W \cup \{x\}$ e per ogni $y \notin W$ aggiorna $e(y)$ e $\rho(y)$ come segue

$$e(y) = \begin{cases} e(y) & \text{se } \rho(y) \leq \rho(x) + d_{xy} \\ x & \text{altrimenti} \end{cases} \qquad \rho(y) = \min\{\rho(y), \rho(x) + d_{xy}\}.$$

4 Ritorna al Passo 1.

Si noti che per ogni $y \in V$, il valore $\rho(y)$ rappresenta a ogni iterazione la lunghezza del cammino minimo da s a y *passando solo attraverso nodi in* W, mentre in $e(y)$ è memorizzato il nodo che precede immediatamente y in tale cammino (per il nodo s, nodo di partenza, si usa l'etichetta $-$ per indicare che non è preceduto da altri nodi).

La prima domanda che possiamo porci riguarda la *correttezza* dell'algoritmo: ci restituisce effettivamente il cammino minimo tra il nodo s e tutti gli altri nodi quando tutte le distanze sono non negative? Vale la seguente osservazione.

Osservazione 26 *Se $d_{ij} \geq 0 \ \forall \ (i,j) \in A$, quando il nodo x viene inserito in W al Passo 3, il valore $\rho(x)$ rappresenta la distanza minima tra s e x. Il cammino minimo può essere ricostruito procedendo a ritroso a partire dall'etichetta $e(x)$.*

Dimostrazione Il valore $\rho(x)$ è la lunghezza del cammino minimo da s a x *passando solo attraverso nodi in* W. Ipotizziamo per assurdo che esista un cammino da s a x di lunghezza inferiore a $\rho(x)$ che passi attraverso nodi $\notin W$ e sia z il primo di tali nodi:

$$s \to \cdots \to z \to \cdots \to x. \tag{10.1}$$

Se interrompiamo tale cammino a z, otteniamo un cammino da s a z con le seguenti caratteristiche:

- passa solo attraverso nodi in W e quindi ha lunghezza non inferiore a $\rho(z)$;
- essendo tutte le distanze non negative, dobbiamo avere che tale cammino ha lunghezza non superiore alla lunghezza del cammino (10.1) e quindi, per ipotesi, strettamente inferiore a $\rho(x)$.

Allora, dobbiamo avere $\rho(z) < \rho(x)$ che però contraddice la regola di scelta di x al Passo 2.

Analizziamo la complessità dell'algoritmo.

Osservazione 27 *L'algoritmo di Dijkstra richiede un numero di operazioni* $O(n^2)$.

Dimostrazione Ci sono n iterazioni. In ciascuna di queste si deve calcolare il minimo tra $\mid V \setminus W \mid \leq n$ valori e per ogni nodo in $V \setminus W$ confrontare due valori. Quindi a ogni iterazione eseguiamo al più $O(n)$ iterazioni. Quindi, complessivamente eseguiamo $O(n^2)$ operazioni.

Quindi l'algoritmo ha complessità polinomiale.

Esempio 32 *Si applichi l'algoritmo di Dijkstra a un grafo* $G = (V, A)$ *con* $V = \{a, b, c, d\}$*, con nodo di partenza* $s \equiv a$ *e con la seguente matrice di distanze:*

$$D = \begin{bmatrix} * & 3 & 12 & 16 \\ 9 & * & 18 & 7 \\ 5 & * & * & 3 \\ 8 & * & 1 & * \end{bmatrix}$$

Iterazione 1 $W = \{a\}$

Nodo	a	b	c	d
ρ	0	3	12	16
e	$-$	a	a	a

Iterazione 2 $x = b \to W = \{a, b\}$

Nodo	a	b	c	d
ρ	0	3	12	10
e	$-$	a	a	b

Iterazione 3 $x = d \to W = \{a, b, d\}$

Nodo	a	b	c	d
ρ	0	3	11	10
e	$-$	a	d	b

Iterazione 4 $x = c \to W = \{a, b, c, d\}$*: STOP.*

Quindi i cammini minimi da a rispettivamente verso b, c *e* d *sono* $a \to b$ *(di lunghezza 3),* $a \to b \to d \to c$ *(di lunghezza 11),* $a \to b \to d$ *(di lunghezza 10).*

Algoritmo di Floyd-Warshall

Definiamo innanzitutto l'operazione di triangolazione.

Definizione 4 *Data una matrice* $n \times n$ *di distanze* R*, per un dato* $j \in \{1, \ldots, n\}$ *chiamiamo* operazione di triangolazione *il seguente aggiornamento della matrice* R*:*

$$R_{ik} = \min\{R_{ik}, R_{ij} + R_{jk}\} \quad \forall\ i, k \in \{1, \ldots, n\} \setminus \{j\}.$$

L'algoritmo di Floyd-Warshall funziona come segue:

Inizializzazione Per $i \neq j$ poni $R_{ij} = d_{ij}$. Poni $R_{ii} = +\infty$ per ogni $i = 1, \ldots, n$. Definisci la matrice $n \times n$ con componenti E_{ij} inizialmente tutte pari a $-$. Poni $j = 1$.

1 Esegui l'operazione di triangolazione con j fissato e aggiorna E come segue

$$E_{ik} = \begin{cases} j & \text{se } R_{ik} > R_{ij} + R_{jk} \\ E_{ik} & \text{altrimenti} \end{cases}$$

2 Se $j = n$ o esiste $R_{ii} < 0$, allora STOP, altrimenti poni $j = j + 1$ e vai al Passo 1.

Nel caso di distanze $d_{ij} \geq 0$, la condizione di arresto $R_{ii} < 0$ non potrà mai verificarsi e si dimostra che i valori R_{ij} danno la lunghezza del cammino minimo da i a j per ogni $i \neq j$, mentre le etichette E_{ij} consentono di ricostruire tali cammini minimi. Nel caso di distanze negative, se non interviene la condizione di arresto $R_{ii} < 0$, allora anche qui gli R_{ij} danno la lunghezza del cammino minimo da i a j; se invece a una certa iterazione si verifica la condizione $R_{ii} < 0$, questa indica la presenza di un ciclo a costo negativo nel grafo. In tal caso, anche ignorando la condizione di arresto $R_{ii} < 0$, non possiamo garantire che al momento della terminazione con $j = n$ gli R_{ij} diano la lunghezza del cammino minimo da i a j.

Prima di vedere un esempio, indichiamo il numero di operazioni dell'algoritmo.

Osservazione 28 *L'algoritmo di Floyd-Warshall richiede un numero di operazioni* $O(n^3)$*.*

Dimostrazione Ci sono n iterazioni. In ciascuna di queste si deve eseguire un'operazione di triangolazione che richiede un numero di operazioni pari a $O(n^2)$. Quindi, complessivamente eseguiamo $O(n^3)$ operazioni.

Quindi anche questo algoritmo ha complessità polinomiale.

Esempio 33 *Si consideri il problema con le seguenti distanze;*

$$D = \begin{bmatrix} * & * & * & 1 \\ 2 & * & 1 & * \\ * & * & * & * \\ * & -4 & 3 & * \end{bmatrix}$$

Inizializzazione

$$R = \begin{bmatrix} \infty & \infty & \infty & 1 \\ 2 & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty \\ \infty & -4 & 3 & \infty \end{bmatrix} \quad E = \begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}$$

Iterazione 1 $j = a$

$$R = \begin{bmatrix} \infty & \infty & \infty & 1 \\ 2 & \infty & 1 & 3 \\ \infty & \infty & \infty & \infty \\ \infty & -4 & 3 & \infty \end{bmatrix} \quad E = \begin{bmatrix} - & - & - & - \\ - & - & - & a \\ - & - & - & - \\ - & - & - & - \end{bmatrix}$$

Iterazione 2 $j = b$

$$R = \begin{bmatrix} \infty & \infty & \infty & 1 \\ 2 & \infty & 1 & 3 \\ \infty & \infty & \infty & \infty \\ -2 & -4 & -3 & -1 \end{bmatrix} \quad E = \begin{bmatrix} - & - & - & - \\ - & - & - & a \\ - & - & - & - \\ b & - & b & b \end{bmatrix}$$

Si ha $R_{dd} < 0$ e quindi l'algoritmo si arresta in quanto è stato individuato un ciclo di lunghezza negativa. Dalla matrice di etichette E il ciclo di lunghezza negativo si identifica in questo modo:

- *il valore $R_{dd} < 0$ indica che il punto di partenza e di arrivo del ciclo è il nodo d*
$$d \to \cdots \to d;$$
- *da $E_{dd} = b$ abbiamo che d nel ciclo è seguito da b*
$$d \to b \to \cdots \to d;$$
- *da $E_{bd} = a$ abbiamo che b nel ciclo è seguito da a*
$$d \to b \to a \to \cdots \to d;$$

- *infine, essendo $E_{ad} = -$, possiamo dire che non ci sono nodi del ciclo tra a e d e quindi chiudere il ciclo*

$$d \to b \to a \to d$$

di lunghezza pari a $R_{dd} = -1$.

Ci si può chiedere se esistono algoritmi di complessità polinomiale in grado di restituire soluzioni ottime del problema in presenza di cicli negativi. In un senso che vedremo in seguito, tale problema risulta essere difficile: non sono noti algoritmi polinomiali per esso e presumbilmente non ne esistono.

10.1.3 Algoritmi per `MST`

Esistono diversi algoritmi di complessità polinomiale per risolvere questo problema.

Algoritmo greedy

L'algoritmo greedy per `MST` è il seguente.

Inizializzazione Si ordinino tutti gli $m = \mid E \mid$ archi del grafo in ordine non decrescente rispetto al peso, cioè

$$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_{m-1}) \leq w(e_m).$$

Si ponga $E_T = \emptyset$. e $k = 1$

1 Se $\mid E_T \mid = \mid V \mid -1$, STOP e si restituisce l'albero $T = (V, E_T)$ come soluzione. Altrimenti si vada al Passo 2.

2 Se e_k non forma cicli con gli archi in E_T, si ponga $E_T = E_T \cup \{e_k\}$. Altrimenti si lasci E_T invariato.

3 Si ponga $k = k + 1$ e si ritorni al Passo 3.

Dimostriamo che tale algoritmo è corretto, ovvero risolve il problema `MST`.

Osservazione 29 *L'algoritmo greedy risolve* `MST`*.*

Dimostrazione Supponiamo per assurdo che esista un albero di supporto $T' = (V, E_{T'})$ a peso minimo con peso inferiore a $T = (V, E_T)$, ovvero quello restituito dall'algoritmo greedy. Indichiamo con e_h l'arco a peso più piccolo tra quelli in $E_T \setminus E_{T'}$. Andiamo ad aggiungere e_h a $E_{T'}$. In tal caso si forma esattamente un ciclo che deve contenere almeno un arco $e_r \notin E_T$ (gli archi in E_T formano un albero di supporto e quindi non possono generare cicli). Si deve avere che $w_{e_r} \geq w_{e_h}$, altrimenti l'algoritmo greedy avrebbe selezionato e_r al posto di e_h (e_r *non* forma cicli con gli archi in E_T selezionati fino al momento in cui viene inserito e_h se e_h è il primo degli archi in E_T che non fanno parte di $E_{T'}$). Ma allora possiamo togliere e_r e sostituirlo con e_h in modo da ottenere

un albero di supporto a peso non superiore rispetto a T'. Iterando questo ragionamento, possiamo sostituire tutti gli archi in $E_{T'} \setminus E_T$ con archi in E_T senza mai aumentare il peso di T' fino a riottenere l'albero T con peso non superiore a T', il che contraddice l'ipotesi iniziale.

Se ora analizziamo la complessità, notiamo che l'operazione più costosa, almeno nel caso di grafi densi, ovvero con un numero di archi $O(\mid V \mid^2)$, è quella di ordinamento degli archi secondo il costo non decrescente. Questo richiede un numero di operazioni pari a $O(\mid E \mid \log(\mid E \mid))$. Il ciclo che segue è di lunghezza non superiore a $\mid E \mid$ e, utilizzando opportune strutture dati, il numero di operazioni da esso richieste non supera $O(\mid E \mid \log(\mid E \mid))$. Abbiamo quindi il seguente risultato.

Osservazione 30 *L'algoritmo greedy ha complessità* $O(\mid E \mid \log(\mid E \mid))$.

Concludiamo con un esempio.

Esempio 34 *Sia dato il grafo completo* $G = (V, E)$ *con* $V = \{a_1, a_2, a_3, a_4\}$ *e pesi*

$$w_{a_1a_2} = 2 \quad w_{a_1a_3} = 3 \quad w_{a_1a_4} = 8 \quad w_{a_2a_3} = 4 \quad w_{a_2a_4} = 7 \quad w_{a_3a_4} = 5$$

L'ordine non decrescente dei pesi degli archi è

$$w_{a_1a_2} \leq w_{a_1a_3} \leq w_{a_2a_3} \leq w_{a_3a_4} \leq w_{a_2a_4} \leq w_{a_1a_4}.$$

Quindi l'algoritmo greedy inserisce (a_1, a_2), (a_1, a_3), *scarta* (a_2, a_3) *e termina con l'inserimento di* (a_3, a_4), *restituendo l'albero di supporto con* $E_T = \{(a_1, a_2)\ (a_1, a_3)\ (a_3, a_4)\}$.

Algoritmo MST-1

Il successivo algoritmo, indicato con MST-1, che vedremo per la risoluzione di `MST` si basa su un risultato presentato qui di seguito. Chiamiamo *foresta di supporto* di un grafo G un grafo parziale $F = (V, E_F)$ di G privo di cicli. In particolare, un albero di supporto è una foresta con una sola componente connessa.

Teorema 7 *Indichiamo con* $(V_1, E_1), \ldots, (V_k, E_k)$ *le componenti connesse di una foresta di supporto* $F = (V, E_F)$ *del grafo* G. *Sia* (u, v) *un arco a peso minimo tra quelli con un solo estremo in* V_1. *Allora, tra tutti gli alberi di supporto a peso minimo tra quelli contenenti* $\cup_{i=1}^{k} E_i$, *ce ne è almeno uno che contiene* (u, v).

Dimostrazione Per assurdo si supponga che ci sia un albero di supporto $T = (V, E_T)$ con $E_T \supseteq \cup_{i=1}^{k} E_i$ e di peso minore rispetto a tutti quelli che contengono (u, v), dove ipotizziamo $u \in V_1$ e $v \notin V_1$. Aggiungiamo (u, v) a E_T. In tal caso si forma esattamente un ciclo che oltre a (u, v), deve contenere anche un altro arco (u', v') con $u' \in V_1$ e $v' \notin V_1$ (altrimenti il ciclo che parte

da $u \in V_1$ non potrebbe chiudersi). Per come è definito (u, v), si deve avere che $w_{uv} \leq w_{u'v'}$. Se ora togliamo (u', v') otteniamo un albero di supporto a peso non superiore a T, che contiene tutti gli archi in $\cup_{i=1}^{k} E_i$ (quindi anch'esso a peso minimo tra gli alberi che contengono tali archi) e che contiene anche (u, v), il che contraddice l'ipotesi iniziale.

Vediamo ora l'algoritmo MST-1.

Inizializzazione Scegli un nodo $v_1 \in V$. Poni $U = \{v_1\}$, $E_T = \emptyset$ e

$$c(v) = v_1 \quad \forall\, v \in V \setminus \{v_1\}.$$

(nel corso dell'algoritmo $c(v)$ conterrà, per i nodi in $V \setminus U$, il nodo in U più vicino a v).

1 Se $U = V$, STOP.

2 Seleziona

$$\bar{v} \in \arg \min_{v \in V \setminus U} w_{vc(v)}$$

($\bar{v}$ è il nodo in $V \setminus U$ più vicino a un nodo in U)

3 Poni $U \cup \{\bar{v}\}$ e $E_T = E_T \cup \{(\bar{v}, c(\bar{v}))\}$.

4 Per ogni $v \in V \setminus U$, se

$$w_{v\bar{v}} < w_{vc(v)}$$

poni $c(v) = \bar{v}$. Ritorna al Passo 1.

Si dimostra il segunete risultato.

Osservazione 31 *L'algoritmo MST-1 è corretto.*

Dimostrazione La dimostrazione si basa sul Teorema 7. Inzialmente abbiamo la foresta con $V_1 \equiv U = \{v_1\}$, $V_i = \{v_i\}$ $i = 2, \ldots, n$, con tutti gli $E_i = \emptyset$. Alla prima iterazione si inserisce l'arco (v_1, v_j), $j \neq 1$, a peso minimo tra quelli con un solo estremo in $U \equiv V_1$ e quindi, in base al Teorema 7, tale arco farà parte dell'albero di supporto a peso minimo tra tutti quelli contenenti $\cup_{i=1}^{n} E_i = \emptyset$ e quindi in realtà l'albero di supporto a peso minimo tra tutti quelli del grafo. Con l'aggiunta di questo arco, le due componenti connesse (V_1, E_1) e (V_j, E_j) si fondono in un'unica componente connessa con nodi $U = \{v_1, v_{j_1}\}$ e l'insieme di archi $E_T = \{(v_1, v_{j_1})\}$, mentre le altre componenti connesse non cambiano. Abbiamo cioè le componenti connesse

$$(U, E_T), \qquad (V_i, \emptyset)\ i \in \{2, \ldots, n\} \setminus \{j_1\}.$$

Alla seconda iterazione andiamo a selezionare il nodo v_{j_2} e il relativo arco $(v_{j_2}, c(v_{j_2}))$ con il peso minimo tra tutti quelli con un solo estremo in U. In base al Teorema 7, l'arco $(v_{j_2}, c(v_{j_2}))$ farà parte di un albero di supporto a peso

minimo *tra tutti quelli che contengono l'unione di tutti gli archi delle componenti connesse*, che si riduce a E_T. Ma poichè E_T contiene il solo arco (v_1, v_{j_1}) che, come dimostrato precedentemente fa parte di un albero di supporto a peso minimo, possiamo anche dire che l'arco aggiunto farà parte di un albero di supporto a peso minimo tra tutti quelli possibili. Quindi andiamo a inserire in U il nodo v_{j_2} e in E_T l'arco $(v_{j_2}, c(v_{j_2}))$ e avremo quindi le nuove componenti connesse

$$(U, E_T), \qquad (V_i, \emptyset)\ i \in \{2, \ldots, n\} \setminus \{j_1, j_2\}.$$

Osservando che l'unione degli archi delle componenti connesse coincide sempre con E_T e che E_T contiene solo archi che fanno parte di un albero di supporto a peso minimo, possiamo iterare il ragionamento garantendo in questo modo che quando $U = V$ (e $\mid E_T \mid = \mid V \mid -1$), (V, E_T) sia un albero di supporto a peso minimo per il grafo G.

Valutiamo ora la complessità dell'algoritmo.

Osservazione 32 *L'algoritmo MST-1 richiede un numero di operazioni* $O(\mid V \mid^2)$.

Dimostrazione Il numero di iterazioni è pari a $\mid V \mid -1$. In ogni iterazione dobbiamo trovare un minimo tra un numero di valori pari a $\mid V \setminus U \mid$ e aggiornare (eventualmente) i valori c per i nodi in $V \setminus U$. Dal momento che $\mid V \setminus U \mid \leq \mid V \mid$, abbiamo complessivamente un numero di operazioni pari a $O(\mid V \mid^2)$.

Si può anche dimostrare che questo algoritmo ha complessità ottima per `MST` almeno per grafi densi. Infatti, per tali grafi non possiamo aspettarci di fare meglio di $O(\mid V \mid^2)$: la sola operazione di lettura dei dati di input (i pesi degli archi) richiede già $O(\mid V \mid^2)$.

Esempio 35 *Nell'esempio visto in precedenza, avremo inizialmente* $U = \{a_1\}$, $E_T = \emptyset$. *Alla prima iterazione avremo* $\bar{v} = a_2$ *e quindi* $U = \{a_1, a_2\}$, $E_T = \{(a_1, a_2)\}$ *con* $c(a_3) = a_1,\ c(a_4) = a_2$. *Alla seconda iterazione avremo* $\bar{v} = a_3$ *e quindi* $U = \{a_1, a_2, a_3\}$, $E_T = \{(a_1, a_2)\ (a_1, a_3)\}$ *con* $c(a_4) = a_3$. *Infine, alla terza iterazione avremo* $\bar{v} = a_4$ *e quindi* $U = \{a_1, a_2, a_3, a_4\} = V$, $E_T = \{(a_1, a_2)\ (a_1, a_3)\ (a_3, a_4)\}$ *e l'algoritmo si arresta.*

Algoritmo MST-2

Inizializzazione Poni $E_T = \emptyset$ e sia $\mathcal{C} = \{S_1, \ldots, S_n\}$ con $S_i = \{v_i\}$ per ogni $i = 1, \ldots, n$ ($\mathcal{C}$ nel corso dell'algoritmo conterrà la collezione di componenti connesse di (V, E_T) ed essendo inizialmente E_T vuoto, viene inizializzata con n componenti, una per ciascun nodo del grafo). Poni

$$componente[v_j] = S_j \quad j = 1, \ldots, n.$$

($componente[v]$ restituisce la componente connessa a cui appartiene il nodo v durante l'algoritmo).

1 Se $\mid \mathcal{C} \mid = 1$, allora STOP.

2 Per ogni $S_i \in \mathcal{C}$, poni $min[i] = \infty$.

3 Per ogni $(u, v) \in E$, siano $S_i = componente[u]$ e $S_j = componente[v]$. Se $i \neq j$, allora:

- $w_{uv} < min[i] \Rightarrow shortest[i] = (u, v)$;
- $w_{uv} < min[j] \Rightarrow shortest[j] = (u, v)$;

(si noti che $shortest[i]$ indica l'arco a peso minimo tra quelli con un solo nodo in S_i).

4 Per tutti gli $S_i \in \mathcal{C}$ poni

$$E_T = E_T \cup \{shortest[i]\}.$$

5 Poni $\mathcal{C}$ pari all'insieme di componenti connesse di (V, E_T) e aggiorna i valori $componente[v]$ per ogni $v \in V$. Torna al Passo 1.

Non dimostriamo la correttezza dell'algoritmo (lo si può fare per esercizio tenendo conto che anche questa è basata sul Teorema 7). Vediamo invece la complessità dell'algoritmo.

Osservazione 33 *L'algoritmo MST-2 richeide un numero di operazioni* $O(\mid E \mid \log(\mid V \mid))$.

Dimostrazione In ogni iterazione dell'algoritmo abbiamo:

- al Passo 3 un ciclo su tutti gli archi del grafo dove per ogni arco dobbiamo eseguire dei confronti e aggiornare (eventualmente) i valori *shortest* per i due nodi dell'arco. Quindi, questo ciclo richiede un numero di operazioni $O(\mid E \mid)$;
- il Passo 4 con l'aggiunta di un numero di archi non superiore a $O(\mid V \mid)$ richiedono un numero di operazioni di ordine non superiore a $O(\mid V \mid)$.

Osservando che lo sforzo per il Passo 4 è dominato da quello per il Passo 3 (i grafi in cui sia presente almeno un albero di supporto si deve avere che il numero di archi non può essere inferiore a $\mid V \mid -1$), in una singola iterazione il numero di operazioni è dell'ordine di $O(\mid E \mid)$.

Chiediamoci ora quante iterazioni vengono eseguite. Ci si arresta quando $\mid \mathcal{C} \mid = 1$. Quello che vogliamo mostrare è che $\mid \mathcal{C} \mid$, inizialmente pari a $\mid V \mid$, viene almeno dimezzato a ogni iterazione. In effetti a ogni iterazione dell'algoritmo una componente connessa contiene *almeno* due componenti connesse dell'iterazione precedente, visto che ogni componente connessa dell'iterazione precedente è unita, con l'aggiunta dell'arco $shortest[j]$, a un'altra componente di tale iterazione. Se a ogni iterazione dimezziamo (almeno) $\mid \mathcal{C} \mid$ arriveremo a

$| \mathcal{C} |= 1$ in al più $\log(| V |)$ iterazioni, da cui il risultato che si voleva dimostrare.

Si noti che per grafi densi con $| E |= O(| V |^2)$ questa complessità è peggiore di quella di MST-1, ma se il numero di archi scende sotto l'ordine $O(| V |^2 / \log(| V |))$ l'algoritmo MST-2 ha prestazioni migliori di MST-1.

Esempio 36 *Nell'esempio visto in precedenza, avremo inizialmente* $\mathcal{C} = \{\{a_1\}\ \{a_2\}\ \{a_3\}\ \{a_4\}\}$, $E_T = \emptyset$. *Alla prima iterazione avremo*

$shortest[1] = (a_1, a_2)$ $shortest[2] = (a_1, a_2)$ $shortest[3] = (a_1, a_3)$ $shortest[4] = (a_3, a_4)$

e quindi avremo l'aggiornamento

$$E_T = \{(a_1, a_2)\ (a_1, a_3)\ (a_3, a_4)\}$$

con una singola componente connessa di (V, E_T), *il che consente di fermare immediatamente l'algoritmo.*

10.1.4 Algoritmi per PL e PLI

In precedenza abbiamo studiato algoritmi per la risoluzione dei problemi dei problemi di PL (simplesso primale e duale) e PLI (algoritmi di taglio e branch-and-bound). Possiamo chiederci quale sia la complessità di questi algoritmi.

Un po' a sorpresa l'algoritmo del simplesso (primale e duale) non ha complessità polinomiale. Va però detto che:

- il comportamento medio dell'algoritmo del simplesso è piuttosto buono, con un nuemro di operazioni dell'ordine del numero dei vincoli del problema;
- le istanze per le quali esso richiede un tempo di esecuzione esponenziale sono molto particolari e difficilmente riscontrabili nella pratica.

Questo mette in luce un possibile limite dell'analisi worst-case: una singola o poche istanze sfortunate su una data dimensione offuscano il buon comportamento dell'algoritmo su tutte le altre istanze con quella dimensione.
Esistono comunque altri algoritmi che risolvono i problemi di PL e che hanno complessità polinomiale (algoritmo dell'elissoide, algoritmi del punto interno). Da un punto di vista pratico il comportamento di tali algoritmi è in realtà molto peggiore (algoritmo dell'elissoide) o confrontabile (algoritmi del punto interno) rispetto all'algoritmo del simplesso.

Per quanto riguarda i problemi di PLI, gli algoritmi visti (e, più genericamente, tutti quelli noti per la PLI) sono di complessità esponenziale. La cosa non è immediatamente visibile per gli algoritmi di taglio, mentre per il branch-and-bound lo si vede più facilmente. Si pensi al caso con sole variabili binarie. Nel caso peggiore, in cui non si riesce a chiudere nessun nodo se non nel momento in cui si sono fissati i valori di tutte le n variabili binarie, l'albero di branch-and-bound si espande fino ad avere un numero di nodi dell'ordine di $O(2^n)$.

10.2 Teoria della complessità

Se fino a ora abbiamo parlato di *problemi di ottimizzazione*, nella teoria della complessità si parla generalmente di *problemi di riconoscimento*. Mentre, come visto, nei problemi di ottimizzazione, data un'istanza del problema, vogliamo restituire in output soluzione ottima e valore ottimo dell'istanza, nei problemi di riconsocimento data un'istanza l'output corrispondente è semplicemente un SI oppure un NO. Vediamo alcuni esempi di problemi di riconoscimento.

`Satisfiability` Date n variabili booleane $x_1, \ldots, x_n$ e una formula booleana in forma normale disgiuntiva (AND di un certo numero m di *clausole*, dove ogni clausola è un OR di variabili booleane eventualmente negate), si vuole stabilire se esiste un assegnamento di valori alle variabili booleane che rende vera la formula.

`3-Satisfiability` Come il problema di `Satisfiability` ma con non più di tre variabili booleane (eventualmente negate) in ogni clausola.

`Circuito hamiltoniano` Dato un grafo $G = (V, E)$, stabilire se esiste un circuito hamiltoniano in tale grafo, ovvero un ciclo che tocca tutti i nodi del grafo una e una sola volta.

Come si vede, in tutti gli esempi, data un'istanza, ci si aspetta come output un SI o un NO.

In realtà sotto ipotesi piuttosto deboli si può mostrare che per ogni problema di ottimizzazione, ne esiste uno corrispondente di riconoscimento di difficoltà analoga. In particolare, consideriamo un problema di ottimizzazione di minimo (il discorso è del tutto analogo per il massimo) con le relative istanze (f, S). Supponiamo che per ogni istanza esistano L e U tali che

$$L \leq f(x) \leq U \quad \forall\, x \in S,$$

ovvero sia possibile definire una limitazione dal di sopra L e una limitazione dal di sopra U della funzione obiettivo f sulla regione ammissibile S e supponiamo anche che tali limitazioni siano legate ai dati del problema attraverso espressioni polinomiali. Supponiamo infine che la funzione obiettivo possa assumere solamente valori interi. Collegata all'istanza del problema di ottimizzazione, definiamo la seguente istanza di un problema di riconoscimento: *dati* (f, S) *e un valore intero* B*, stabilire se esiste un* $x \in S$ *tale che* $f(x) \leq B$. Vediamo alcuni esempi.

`CLIQUE` *riconoscimento*: dato un intero positivo $k \leq \mid V \mid$, stabilire se nel grafo esiste una clique di cardinalità almeno pari a k. In questo caso possiamo scegliere $L = 1$ e $U = \mid V \mid$.

`TSP` **- Traveling Salesman Problem** *riconoscimento*: dato un intero B, stabilire se esiste nel grafo un circuito hamiltoniano di distanza totale non superiore a B. In questo caso possiamo scegliere $L = \mid V \mid \min_{(i,j)\in E} d_{ij}$ e $U = \mid V \mid \max_{(i,j)\in E} d_{ij}$.

`KNAPSACK` *riconoscimento* dato un intero B, stabilire se esiste un sottinsieme K degli n oggetti con peso complessivo $\sum_{i \in K} p_i$ non superiore a b e valore complessivo $\sum_{i \in K} v_i$ non inferiore a B. In questo caso possiamo scegliere $L = 0$ e $U = n \max_i v_i$.

È del tutto evidente che, se siamo in grado di risolvere il problema di ottimizzazione, allora abbiamo anche in modo immediato una risposta al problema di riconoscimento. Infatti, sia $x^* \in S$ la soluzione ottima del problema di ottimizzazione. Evidentemente, si ha che la risposta all'istanza del problema di riconoscimento è SI se e solo se $f(x^*) \leq B$. Ma supponiamo ora di saper risolvere il problema di riconoscimento e consideriamo la seguente procedura:

Binary search

1 si ponga $\mathcal{L}_0 = L$ e $\mathcal{U}_0 = U$, $k = 0$;

2 si ponga $B = \frac{\mathcal{L}_0 + \mathcal{U}_0}{2}$;

3 se il problema di riconoscimento con il valore B fissato al Passo 2. ha risposta SI, allora si ponga $\mathcal{U}_{k+1} = B$, altrimenti si ponga $\mathcal{L}_{k+1} = B$;

4 se $\mathcal{U}_{k+1} - \mathcal{L}_{k+1} < 1$, allora STOP: il valore $\lfloor \mathcal{U}_{k+1} \rfloor$ è il valore ottimo del problema; altrimenti si ponga $k = k + 1$ e si ritorni al Passo 1.

In un numero di iterazioni pari al più a $\log(U - L)$ (e quindi polinomiale rispetto alla dimensione dell'istanza) questa procedura di binary search restituisce il valore ottimo del problema di ottimizzazione. In altre parole, siamo in grado di risolvere in tempo polinomiale il problema di riconoscimento se e solo se siamo in grado di risolvere in tempo polinomiale anche il problema di ottimizzazione.

Per cercare di distinguere tra problemi facili e difficili, la teoria della complessità introduce delle classi di problemi. In particolare qui discuteremo le classi di problemi in $\mathcal{P}$, in $\mathcal{NP}$ e $\mathcal{NP}$-completi.

10.2.1 La classe $\mathcal{P}$

Definizione 5 *Dato un problema di riconoscimento R, diciamo che questo appartiene alla classe $\mathcal{P}$ se e solo se esiste un algoritmo $\mathcal{A}$ di complessità polinomiale che lo risolve.*

Alla classe $\mathcal{P}$ appartengono i problemi `MST` e `SHORTEST PATH` per cui abbiamo presentato più di un algoritmo di complessità polinomiale. Vi appartengono anche i problemi di PL, anche se gli algoritmi di complessità polinomiale che li risolvono (elissoide, punto interno) non sono stati visti. Dall'appartenenza di PL a $\mathcal{P}$ segue l'appartenenza a $\mathcal{P}$ anche di tutte le sottoclassi di problemi di PLI risolvibili come problemi di PL (si veda più avanti l'Osservazione 38).

10.2.2 La classe NP

Definizione 6 *La classe NP contiene tutti i problemi di riconoscimento per i quali, in corrispondenza di un'istanza la cui risposta è affermativa, esiste un* certificato *noto il quale la risposta affermativa può essere data in tempo polinomiale.*

Vediamo alcuni esempi.

`CLIQUE` In tal caso il certificato è una lista di nodi, di cardinalità k, che formano una clique; nota tale lista, verificare che questi nodi formano una clique richiede un tempo $O(k^2)$ (verifica che esiste un arco del grafo per ogni coppia di nodi della lista).

`TSP` Il certificato è un circuito hamiltoniano

$$i_1 \to \cdots \to i_n \to i_{n+1} \equiv i_1$$

con distanza $\sum_{j=1}^{n} d_{i_j i_{j+1}}$ non superiore a B. Tale verifica richiede un tempo $O(n)$ (somma delle n distanze).

`Satisfiability` Il certificato è un assegnamento di valori alle n variabili booleane che rende vere tutte le m clausole.

`Circuito hamiltoniano` Il certificato è una sequenza di n nodi

$$i_1 \to \cdots \to i_n \to i_{n+1} \equiv i_1$$

per cui si verifica che $(i_j, i_{j+1}) \in E$, $j = 1, \ldots, n$.

Va notato che tutti i problemi in $\mathcal{P}$ fanno parte anche di $\mathcal{NP}$, ovvero $\mathcal{P} \subseteq \mathcal{NP}$. Per i problemi in $\mathcal{P}$ si ha infatti che non è neppure necessario esibire un certificato per avere una risposta affermativa in tempo polinomiale: è sufficiente utilizzare un algoritmo di complessità polinomiale per questi problemi (che esiste per definizione di classe $\mathcal{P}$).

10.2.3 Riduzioni in tempo polinomiale

Definizione 7 *Dati due problemi di riconoscimento R_1 e R_2, diciamo che R_1 è riducibile in tempo polinomiale a R_2 se e solo se esiste un algoritmo $\mathcal{A}_1$ di complessità polinomiale per R_1 che richiama (per una o più volte) come sottoprocedura* a costo unitario *un algoritmo $\mathcal{A}_2$ per R_2.*

Vale la seguente osservazione.

Osservazione 34 *Se R_1 è riducibile in tempo polinomiale a R_2 e R_2 è risolvibile in tempo polinomiale, allora anche R_1 è risolvibile in tempo polinomiale.*

Dimostrazione Sia k la dimensione di un'istanza di problema R_1. Sia $p_1(k)$ il numero polinomiale di operazioni dell'algoritmo $\mathcal{A}_1$ ipotizzando che il costo di una chiamata dell'algoritmo $\mathcal{A}_2$ sia di una sola operazione. Il numero di tali chiamate è limitato dal di sopra proprio da $p_1(k)$ (nell'ipotesi in cui ogni singola operazione di $\mathcal{A}_1$ sia proprio una chiamata di $\mathcal{A}_2$). In ogni chiamata di $\mathcal{A}_2$ la dimensione dell'istanza di R_2 da risolvere non potrà essere superiore a $p_1(k)$ (nell'ipotesi che tutte le operazioni di $\mathcal{A}_1$ siano di ricopiatura dell'input per una chiamata di $\mathcal{A}_2$). Se indichiamo con $p_2(h)$ il numero (polinomiale) di operazioni richiesto dall'algoritmo $\mathcal{A}_2$ per risolvere un'istanza di dimensione h di R_2, allora avremo che il numero di operazioni richiesto per risolvere l'istanza di R_1 di dimensione k è non superiore a

$$p_1(k)p_2(p_1(k))$$

che è un'espressione polinomiale, come si voleva dimostrare.

10.2.4 I problemi NP-completi

Definizione 8 *Diciamo che un problema R è $\mathcal{NP}$-completo se:*

- $R \in \mathcal{NP}$*;*
- *per ogni problema $Q \in \mathcal{NP}$, esiste una riduzione polinomiale di Q in R.*

Si noti che la definizione di problema $\mathcal{NP}$-completo unita all'Osservazione 34, implica che se esistesse un problema $\mathcal{NP}$-completo risolvibile in tempo polinomiale (o, equivalentemente, appartenente a $\mathcal{P}$), allora si dovrebbe avere che $\mathcal{P} = \mathcal{NP}$. Abbiamo visto che $\mathcal{P} \subseteq \mathcal{NP}$ e possiamo chiederci ora se $\mathcal{P} = \mathcal{NP}$, oppure se esistono problemi in NP che non sono risolvibili in tempo polinomiale, cioè $\mathcal{P} \neq \mathcal{NP}$. In realtà con le conoscenze attuali non si può ancora rispondere a tale domanda. Tuttavia tra le due possibili risposte quella che si ritiene la più probabile è che $\mathcal{P} \neq \mathcal{NP}$. In base alla definizione di problemi $\mathcal{NP}$-completi e al fatto che non sia chiaro se $\mathcal{P} = \mathcal{NP}$ oppure no, possiamo dire che non sono *al momento* noti algoritmi di complessità polinomiale in grado di risolvere problemi $\mathcal{NP}$-completi. Inoltre, se, come si ritiene, $\mathcal{P} \neq \mathcal{NP}$, allora non esisterebbero algoritmi di complessità polinomiale per tali problemi. Ne risulta quindi che la classe dei problemi $\mathcal{NP}$-completi è una classe di problemi difficili nel senso che, a meno che non sia $\mathcal{P} = \mathcal{NP}$, non possiamo risolverli in tempo polinomiale. Esiste un gran numero di problemi $\mathcal{NP}$-completi. La dimostrazione di $\mathcal{NP}$-completezza di un problema si basa spesso sulla seguente osservazione.

Osservazione 35 *Se R_1 è $\mathcal{NP}$-completo e può essere ridotto in tempo polinomiale a $R_2 \in \mathcal{NP}$, allora anche R_2 è $\mathcal{NP}$-completo.*

Dimostrazione Segue dalla proprietà transitiva delle riduzioni polinomiali: se un generico problema $Q \in \mathcal{NP}$ è riducibile in tempo polinomiale a R_1 e a sua volta R_1 è riducibile in tempo polinomiale a R_2, ne consegue che Q è riducibile in tempo polinomiale a R_2.

Naturalmente, per poter sfruttare questo risultato, è necessario che sia già noto qualche problema $\mathcal{NP}$-completo. Per esempio, si può dimostrare (ma noi ometteremo tale dimostrazione) il seguente risultato

Osservazione 36 *I problemi* `Satisfiability` *e* `3-Satisfiability` *sono $\mathcal{NP}$-completi.*

Tale risultato può essere utilizzato per dimostrare la $\mathcal{NP}$-completezza del problema `CLIQUE`.

Osservazione 37 *Il problema* `CLIQUE` *è $\mathcal{NP}$-completo.*

Dimostrazione Consideriamo la seguente trasformazione polinomiale di un problema di `3-Satisfiability` in un problema `CLIQUE`. Definiamo un *assegnamento di verità parziale* come un assegnamento di valore T (true) o F (false) a certe variabili booleane, mentre il valore di altre variabili non viene assegnato e a esse viene attribuita un'etichetta u.
Dato un problema di `3-Satisfiability` lo trasformiamo ora in un problema `CLIQUE` su un grafo $G = (V, E)$ con i seguenti nodi e archi.

Nodi Per ogni clausola C_i creiamo 7 nodi in V, ciascuno dei quali corrisponde a un assegnamento parziale dove le etichette di tutte le $n-3$ variabili *non* coinvolte nella clausola C_i sono fissate a u, mentre alle 3 variabili coinvolte in questa clausola vengono assegnate le 7 possibili combinazioni di valori T e F che rendono la clausola vera (viene esclusa la sola combinazione di valori che rende falsa la clausola). In tal modo il numero di nodi in V è pari a $7m$ (dove m è il numero di clausole).

Archi Uniamo due nodi con un arco se l'assegnamento parziale associato a un nodo è *compatibile* con quello dell'altro nodo, ovvero se per ogni variabile x_i si ha che vale una delle seguenti condizioni:

- ha etichetta u in almeno uno dei due nodi;
- ha valore T in entrambi i nodi;
- ha valore F in entrambi i nodi.

In particolare, si noti che i 7 nodi associati a una stessa clausola non possono essere collegati tra loro da archi.

Ora si dimostra che `3-Satisfiability` ha risposta affermativa se e solo se il grafo costruito ha una clique di dimensione m.
Infatti, supponiamo dapprima che il grafo abbia una clique di dimensione m. In base a quanto osservato, per ogni clausola C_i tale clique deve contenere uno e

un solo nodo associato a questa clausola. Se mettiamo assieme gli assegnamenti parziali (tra loro compatibili) dei nodi della clique, otteniamo un assegnamento completo che soddisfa tutte le m clausole.
Viceversa, supponiamo di avere un assegnamento completo che soddisfa tutte le m clausole. Nel grafo, per ogni clausola C_i andiamo a prendere il nodo relativo all'unico assegnamento parziale compatibile con quello completo dato. L'insieme di questi m nodi forma una clique di dimensione m.

Un'altra possibile dimostrazione di $\mathcal{NP}$-completezza per un problema consiste nel far vedere che contiene una sottoclasse di problemi $\mathcal{NP}$-completi. Vediamo il seguente esempio.

Teorema 8 *I problemi PLI sono $\mathcal{NP}$-completi.*

Dimostrazione Molti dei problemi visti sino a ora possono essere formulati come problemi di PLI. Consideriamo, ad esempio, `CLIQUE` che abbiamo dimostrato essere $\mathcal{NP}$-completo. Il seguente è un modello matematico valido per un problema di clique di cardinalità massima su un grafo $G = (V, E)$. Associamo una variabile binaria x_i a ogni nodo $i \in V$ con

$$x_i = \begin{cases} 0 & \text{se } i \text{ non è inserito nella clique} \\ 1 & \text{altrimenti} \end{cases}$$

Per avere che $C = \{i \ : \ x_i = 1\}$ sia una clique dovremo imporre i seguenti vincoli

$$x_i + x_j \leq 1 \quad \forall\ (i,j) \notin E,\ i \neq j$$

(in pratica, imponiamo che un solo tra i nodi i e j faccia parte della clique se i due nodi non sono collegati da un arco). La cardinalità della clique C è data semplicemente da $\sum_{i=1}^{|V|} x_i$. Quindi arriviamo al modello matematico

$$\begin{array}{ll} \max & \sum_{i=1}^{|V|} x_i \\ & x_i + x_j \leq 1 \quad \forall\ (i,j) \notin E,\ i \neq j \\ & x_i \in \{0,1\} \quad i = 1, \ldots, \mid V \mid \end{array}$$

Avendo quindi dimostrato che ogni istanza di problema `CLIQUE` può essere vista come un'istanza di problema di PLI (più precisamente, di problema di PL binaria) con un numero di variabili e di vincoli polinomiale rispetto alla dimensione dell'istanza di `CLIQUE`, la $\mathcal{NP}$-completezza di `CLIQUE` ci permette di concludere che anche PLI è $\mathcal{NP}$-completo.

Abbaimo visto che la $\mathcal{NP}$-completezza di una sottoclasse implica la $\mathcal{NP}$-completezza della classe stessa. Viceversa non possiamo affermare che la $\mathcal{NP}$-completezza di una classe implichi la $\mathcal{NP}$-completezza delle sue sottoclassi. Per esempio, abbiamo il seguente risultato.

Osservazione 38 *Le seguenti sottoclassi dei problemi di PLI:*

- *flusso a costo minimo*

- *flusso massimo*
- *trasporto*
- *assegnamento*

appartengono a $\mathcal{P}$.

Dimostrazione Sulla base dell'Osservazione 22 tutte queste sottoclassi sono risolvibili come problemi di PL e quindi in tempo polinomiale.

Concludiamo ora la parte sui problemi $\mathcal{NP}$-completi, citando (senza dimostrazione) i risultati di $\mathcal{NP}$-completezza per altri problemi precedentemente introdotti.

Osservazione 39 *I problemi* `KNAPSACK`*,* `TSP`*,* `Circuito hamiltoniano` *sono $\mathcal{NP}$-completi.*

10.2.5 Problemi e algoritmi di approssimazione

Sia data una classe di problemi di ottimizzazione tale che per ogni istanza (f, S) si ha

$$\forall\ x \in S\ :\quad f(x) \geq 0.$$

Indichiamo con $opt_{f,S}$ il valore ottimo dell'istanza. Il problema di ε-approssimazione associato al problema di ottimizzazione è definito come segue.

Definizione 9 *Per i problemi di massimo il problema di ε-approssimazione, $\varepsilon \geq 0$, consiste nel determinare un punto $\overline{x} \in S$ tale che*

$$\frac{opt_{f,S}}{f(\overline{x})} \leq 1 + \varepsilon. \tag{10.2}$$

Per i problemi di minimo il problema di ε-approssimazione consiste nel determinare un punto $\overline{x} \in S$ tale che

$$\frac{f(\overline{x})}{opt_{f,S}} \leq 1 + \varepsilon. \tag{10.3}$$

In entrambi i casi il punto $\overline{x}$ viene definito soluzione ε-approssimata del problema.

Si noti che per $\varepsilon = 0$ il problema coincide con quello di ottimizzazione, ma per $\varepsilon > 0$ si richiede qualcosa di meno rispetto al problema di ottimizzazione: non si cerca la soluzione ottima ma una soluzione che non si discosti troppo da quella ottima. In particolare, da (10.2) e (10.3) si ricava che in una soluzione ε-approssimata il valore f in corrispondenza di tale soluzione differisce, sia per i problemi di massimo che per quelli di minimo, per al più $\varepsilon opt_{f,S}$ dal valore ottimo $opt_{f,S}$ dell'istanza. Chiaramente, tanto maggiore è il valore di ε, quanto minore è la precisione garantita da una soluzione ε-approssimata.
Un algoritmo $\mathcal{A}_\varepsilon$ si definisce *algoritmo di ε-approssimazione* per una classe di

problemi di ottimizzazione, se risolve il problema di ε-approssimazione associato al problema di ottimizzazione. Cioè se indichiamo con $\overline{x} \in S$ la soluzione restituita dall'algoritmo $\mathcal{A}_\varepsilon$ e con $A_{f,S} = f(\overline{x})$ il corrispondente valore della funzione obiettivo f, si ha che

$$\frac{opt_{f,S}}{A_{f,S}} \leq 1 + \varepsilon,$$

se il problema è di massimo, oppure

$$\frac{A_{f,S}}{opt_{f,S}} \leq 1 + \varepsilon,$$

se il problema è di minimo. A questo punto ci si può chiedere, dato un problema $\mathcal{NP}$-completo, qual è la complessità dei corrispondenti problemi di ε-approssimazione per diversi possibili valori di ε. Possiamo riconoscere quattro diversi possibili casi che elenchiamo ora in ordine crescente di difficoltà.

Caso 1 Per ogni $\varepsilon > 0$ esiste un algoritmo di ε-approssimazione che richiede tempi polinomiali sia rispetto alla dimensione delle istanze, sia rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. In tal caso si dice che il problema ammette uno *schema di approssimazione completamente polinomiale* o FPTAS (Fully Polynomial Time Appproximation Scheme).

Caso 2 Per ogni $\varepsilon > 0$ esiste un algoritmo di ε-approssimazione che richiede tempo polinomiale rispetto alla dimensione delle istanze ma esponenziale rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. In tal caso si dice che il problema ammette uno *schema di approssimazione polinomiale* o PTAS.

Caso 3 Per valori piccoli di ε, anche il problema di ε-approssimazione è $\mathcal{NP}$-completo, mentre per valori di ε più elevati è risolvibile in tempo polinomiale.

Caso 4 Per ogni valore di ε il problema di ε-approssimazione è $\mathcal{NP}$-completo.

Il problema `KNAPSACK` rientra nel Caso 1, cioè anche se la determinazione di una soluzione ottima del problema `KNAPSACK` è un problema $\mathcal{NP}$-completo, è sempre possibile determinare una soluzione ε-approssimata per ogni $\varepsilon > 0$ tramite un algoritmo polinomiale rispetto alla dimensione dell'istanza e rispetto a $\frac{1}{\varepsilon}$. All'estremo opposto, cioè nel Caso 4, troviamo il problema `TSP`, per il quale, come dimostreremo più sotto, non è possibile risolvere in tempi polinomiali, a meno che non sia $\mathcal{P} = \mathcal{NP}$, neppure il problema di ε-approssimazione per *ogni* valore di ε. Possiamo quindi dire che, pur appartenendo entrambi i problemi di `KNAPSACK` e di `TSP` alla classe dei problemi difficili o, più precisamente, $\mathcal{NP}$-completi, il problema `KNAPSACK` e quello `TSP` hanno livelli di difficoltà agli estremi opposti all'interno della classe dei problemi $\mathcal{NP}$-completi. Per quanto riguarda il Caso 3, in questo rientra la sottoclasse del problema `TSP`, chiamata problema `TSP` *metrico*, dove ricordiamo che le distanze sono simmetriche ($d_{ij} = d_{ji}$ per ogni i, j) e soddisfano la disuguaglianza triangolare $d_{ij} \leq d_{ik} + d_{kj}$ per ogni

i, j, k distinti tra loro. Non vedremo invece esempi di problemi che rientrano nel Caso 2.
Dimostriamo ora l'appartenenza di `TSP` al Caso 4.

Osservazione 40 *A meno che non sia $\mathcal{P} = \mathcal{NP}$, il problema di ε-approssimazione per* `TSP` *non è risolvibile in tempo polinomiale per ogni $\varepsilon > 0$.*

Dimostrazione Ragioniamo per assurdo e supponiamo che per un certo $\varepsilon > 0$ esista un algoritmo di approssimazione $\mathcal{A}_\varepsilon$ per `TSP` di complessità polinomiale. Facciamo vedere che in tal caso si avrebbe che esiste un algoritmo di complessità polinomiale anche per il problema `Circuito hamiltoniano`, che sappiamo essere $\mathcal{NP}$-completo (Osservazione 39). Infatti, dato un grafo $G = (V, E)$ per il quale vogliamo stabilire se esiste un circuito hamiltoniano, costruiamo l'istanza del problema `TSP` con le seguenti distanze degli archi:

$$d_{ij} = \begin{cases} 1 & \text{se } (i,j) \in E \\ 2 + \varepsilon \mid V \mid & \text{altrimenti} \end{cases}$$

Si dimostra che l'algoritmo di approssimazione $\mathcal{A}_\varepsilon$ restituisce una soluzione con valore $\mid V \mid$ se e solo se G ha un circuito hamiltoniano. Infatti, se $\mathcal{A}_\varepsilon$ restituisce una soluzione con valore $\mid V \mid$ questa deve essere necessariamente formata da archi con distanza pari a 1 e quindi archi in E che formano un circuito hamiltoniano. Supponiamo ora invece, per assurdo, che $\mathcal{A}_\varepsilon$ restituisca una soluzione con valore $>\mid V \mid$ e che G contenga un circuito hamiltoniano. In questo caso il circuito hamiltoniano in G ci fornisce una soluzione ottima con valore $\mid V \mid$ dell'istanza del `TSP`, mentre la soluzione restuita da $\mathcal{A}_\varepsilon$ deve avere valore almeno pari a

$$1 + (1 + \varepsilon) \mid V \mid$$

(nel migliore dei casi tale soluzione contiene un solo arco $\notin E$ e quindi a distanza pari a $2 + \varepsilon \mid V \mid$). Ma per tale soluzione si ha

$$\frac{1 + (1 + \varepsilon) \mid V \mid - \mid V \mid}{\mid V \mid} = \frac{1}{\mid V \mid} + \varepsilon > \varepsilon,$$

il che contraddice il fatto che la soluzione restituita da $\mathcal{A}_\varepsilon$ sia ε-approssimata.

Capitolo 11

Algoritmi per problemi di flusso

In questo capitolo presenteremo i dettagli di algoritmi per i problemi di flusso a costo minimo e flusso massimo. Nel Capitolo 2 abbiamo ricavato i modelli matematici di questi problemi, mentre l'Osservazione 38 ci permette di dire che tali problemi, pur essendo sottoclassi dei problemi di PLI, sono risolvibili come problemi di PL rimuovendo i vincoli di interezza sulle variabili. Quindi, sono risolvibili in tempo polinomiale. Inoltre, come già commentato, per le sottoclassi di problemi di PLI risolvibili rimuovendo semplicemente la condizione di interezza sulle variabili, è spesso possibile utilizzare algoritmi già noti per la PL (simplesso con le sue varianti) implementati però in modo tale da sfruttare la particolare struttura di questi problemi. È il caso anche dell'algoritmo che vedremo per il flusso a costo minimo, che sarà una specializzazione del simplesso primale, e dell'algoritmo di Ford-Fulkerson per il flusso massimo, che può essere visto come specializzazione del simplesso primale-duale, una variante del simplesso non trattata in questi appunti.

11.1 Flusso a costo minimo

Facciamo dapprima un'osservazione relativa al rango della matrice dei vincoli di un problema di flusso a costo minimo. Come già osservato nel Capitolo 2, la matrice dei vincoli di uguaglianza di un problema di flusso a costo minimo è la matrice di incidenza nodo-arco di una rete. Se ora sommiamo tra loro tutte le $| V |$ righe della matrice otteniamo il vettore nullo. Infatti in ogni colonna ci sono esattamente un +1 e un -1. Quindi le $| V |$ righe sono tra loro linearmente dipendenti e il rango non potrà essere superiore a $| V | -1$. Si può dimostrare che il rango è esattamente pari a $| V | -1$. Il fatto che $\sum_{i \in V} b_i = 0$ (e quindi non solo le righe della matrice sono linearmente dipendenti ma anche le equazioni stesse dei vincoli sono tra loro linearmente dipendenti) ci mostra che uno (e un solo) vincolo del problema può essere eliminato in quanto ridondante. Non

importa quale vincolo si elimina. Come convenzione si può fissare di eliminare l'ultima equazione. Nel seguito quindi l'ultimo vincolo si intenderà soppresso e quando si parlerà di matrice dei vincoli si intenderà la matrice di incidenza nodo-arco privata dell'ultima riga.

Esempio 37 *Sia data la rete in Figura 2.1. Si ricordi che i valori b_i sono riportati di fianco ai nodi mentre lungo gli archi sono riportati i valori c_{ij}. I nodi 1,2 e 3 sono nodi sorgente mentre i nodi 4 e 5 sono nodi destinazione (non vi sono nodi transito). Il problema corrispondente è il seguente*

$$\begin{gathered}
\min\ 5x_{12} - 4x_{23} + 6x_{42} - 2x_{13} + 0x_{34} + 2x_{15} + 4x_{53} + 3x_{45} \\
x_{12} + x_{13} + x_{15} = 2 \\
x_{23} - x_{12} - x_{42} = 5 \\
x_{34} - x_{13} - x_{23} - x_{53} = 1 \\
x_{42} + x_{45} - x_{34} = -4 \\
x_{53} - x_{15} - x_{45} = -4 \\
x_{12}, x_{23}, x_{42}, x_{13}, x_{34}, x_{15}, x_{53}, x_{45} \geq 0 \text{ interi}
\end{gathered}$$

La matrice di incidenza nodo-arco è la seguente:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{11.1}$$

e quindi, una volta soppressa l'ultima equazione, la matrice dei vincoli sarà

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}$$

ovvero la matrice (11.1) in cui è stata soppressa l'ultima riga.

Vedremo ora come il simplesso primale possa essere adattato ai problemi di flusso a costo minimo su reti. Cominceremo con il discutere un risultato relativo alle basi nei problemi di flusso a costo minimo.

11.1.1 Basi per il problema di flusso a costo minimo

Vogliamo mostrare il legame esistente tra basi del simplesso e alberi di supporto della rete. Si noti che, essendo il numero di righe (e il rango) della matrice dei vincoli pari a $\mid V \mid -1$, le basi sono sempre formate da $\mid V \mid -1$ variabili. Ma non tutti gli aggregati di $\mid V \mid -1$ variabili danno origine a una base. Per formare una base devono anche soddisfare la proprietà che la matrice ottenuta considerando le sole colonne relative a esse nella matrice dei vincoli sia invertibile. Nel nostro

esempio, se si considera l'albero di supporto in Figura 11.1 si ha che le colonne a esso relativo nella matrice dei vincoli formano la seguente matrice

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

che è invertibile e quindi le variabili corrispondenti formano una base. Ora, si

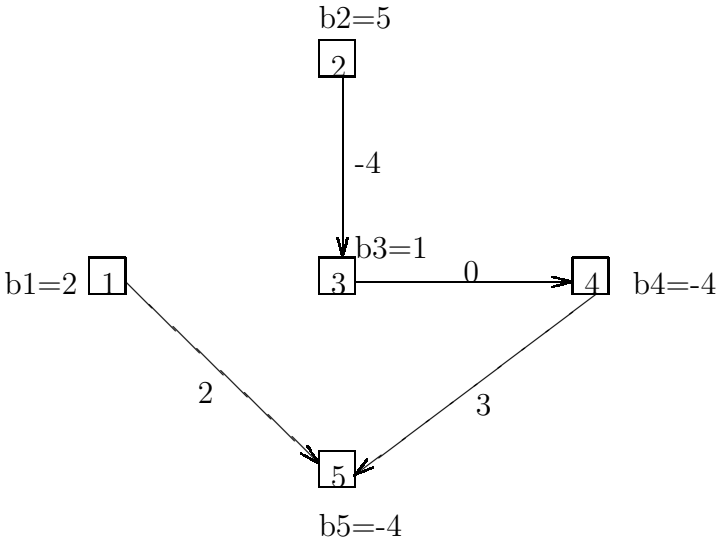


Figura 11.1: Un albero di supporto per la rete del nostro esempio.

consideri un generico albero di supporto della rete. Esso sarà formato da $\mid V \mid -1$ archi. Si può dimostrare (ma non lo faremo) che la soluzione ottenuta mettendo le variabili relative agli archi dell'albero di supporto in base e tutte le altre fuori base si ottiene una soluzione di base (non necessariamente ammissibile e quindi non necessariamente un vertice) del problema. Abbozzeremo invece una dimostrazione del viceversa e cioè che data una qualsiasi base, i $\mid V \mid -1$ archi relativi alle variabili in base formano un albero di supporto. Per dimostrarlo ragioniamo per assurdo e supponiamo che gli archi non formino un albero di supporto. Poiché gli archi sono $\mid V \mid -1$, se non formano un albero di supporto devono formare almeno un ciclo. Vediamo cosa succede in presenza di un ciclo sul nostro esempio, precisando che quanto vedremo su tale esempio può essere generalizzato a tutti i casi in cui compaia un ciclo. Supponiamo che le $\mid V \mid -1 = 4$ variabili in base siano quelle relative agli archi

$$(1,2)\ (2,3)\ (5,3)\ (1,5)$$

che formano il ciclo mostrato in Figura 11.2. Fissiamo un arco del ciclo, ad

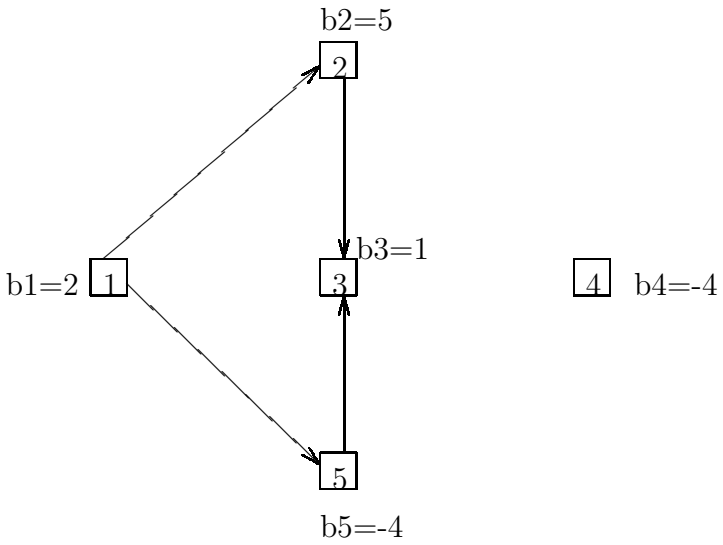


Figura 11.2: Un insieme di archi che formano un ciclo non possono dare origine a una base.

esempio $(1,2)$ ed imponiamo che il verso di percorrenza del ciclo sia quello dell'arco $(1,2)$. Per ogni colonna nella matrice dei vincoli relativa a un arco del ciclo la moltiplichiamo per +1 se il ciclo attraversa l'arco nel suo verso, per -1 se lo attraversa nel verso opposto. Poi sommiamo i vettori ottenuti in questo modo. Nel nostro caso moltiplicheremo per +1 le colonne relative agli archi $(1,2)$ e $(2,3)$ e per -1 quelle relative agli archi $(5,3)$ e $(1,5)$. Quindi avremo

$$+1\begin{bmatrix}1\\-1\\0\\0\end{bmatrix}+1\begin{bmatrix}0\\1\\-1\\0\end{bmatrix}-1\begin{bmatrix}0\\0\\-1\\0\end{bmatrix}-1\begin{bmatrix}1\\0\\0\\0\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

Ciò dimostra che esiste una combinazione lineare non nulla delle colonne che restituisce il vettore nullo. Quindi, tali colonne non formano una matrice invertibile e non rappresentano una base. Come detto, è possibile generalizzare questo risultato: ogni qualvolta gli archi relativi a un insieme di variabili formano un ciclo, le corrispondenti colonne della matrice dei vincoli sono linearmente dipendenti e quindi le variabili non formano una base. L'unica possibilità per avere una base è che gli archi non formino alcun ciclo. Ma in base alla definizione di albero, in un grafo con $\mid V \mid$ nodi, $\mid V \mid -1$ archi che non formino alcun ciclo rappresentano un albero di supporto.
Abbiamo quindi mostrato il seguente importante risultato.

Osservazione 41 *In un problema di flusso su rete a costo minimo vi è una corrispondenza uno a uno tra basi e alberi di supporto, ovvero a ogni insieme*

di $\mid V \mid -1$ *variabili che formano una base corrisponde un albero di supporto e viceversa.*

Quindi, per i problemi di flusso su reti a costo minimo sarà indifferente parlare di basi o di alberi di supporto.

11.1.2 Alberi di supporto e soluzione di base corrispondente

Supponiamo ora di avere un albero di supporto (vedi Figura 11.1) nel nostro esempio e poniamoci la seguente domanda: in corrispondenza di tale albero e quindi di tale soluzione di base, qual è il valore delle variabili? Per prima cosa le variabili associate ad archi che non appartengono all'albero di supporto avranno associato un valore pari a 0. Quindi nel nostro esempio:

$$x_{12} = x_{13} = x_{42} = x_{53} = 0.$$

A questo punto sostituiamo tali valori nulli nei vincoli del problema. Quello che si ottiene è un sistema di $\mid V \mid -1$ variabili (quelle relative agli archi dell'albero di supporto) e $\mid V \mid -1$ equazioni. L'unica soluzione di tale sistema fornisce i valori delle variabili in base. Nel nostro esempio ponendo a 0 i valori delle variabili relative ad archi che non appartengono all'albero di supporto, otteniamo il seguente sistema:

$$\begin{aligned} x_{15} &= 2 \\ x_{23} &= 5 \\ x_{34} - x_{23} &= 1 \\ x_{45} - x_{34} &= -4 \end{aligned}$$

la cui soluzione è:

$$x_{15} = 2 \quad x_{23} = 5 \quad x_{34} = 6 \quad x_{45} = 2$$

Quindi la soluzione relativa all'albero di supporto dell'esempio è data da

$$x_{15} = 2 \quad x_{23} = 5 \quad x_{34} = 6 \quad x_{45} = 2 \quad x_{12} = x_{13} = x_{42} = x_{53} = 0$$

Si noti che tutte le variabili sono non negative e quindi in questo caso si parla di soluzione di base o albero di supporto *ammissibile* (e quindi si tratta di un vertice della regione ammissibile).
NOTA BENE Nel caso in cui una o più delle variabili relative all'albero di supporto fossero uguali a 0 avremmo una soluzione *degenere*.

11.1.3 Calcolo dei coefficienti di costo ridotto

Una condizione sufficiente per stabilire se, data una soluzione di base ammissibile (un vertice), ci troviamo in una soluzione ottima in un problema di PL, è controllare se tutti i coefficienti di costo ridotto sono tutti non positivi in un

problema di massimo oppure tutti non negativi in un problema di minimo. Nella riformulazione rispetto a una base nel simplesso i coefficienti di costo ridotto sono i coefficienti delle variabili fuori base nell'obiettivo. Nel simplesso su rete non abbiamo alcuna riformulazione ripetto alle basi e dobbiamo quindi vedere come calcolare tali valori. Per prima cosa ricordiamo che vanno calcolati per le sole variabili fuori base. Quindi, i coefficienti vanno calcolati per le sole variabili associate ad archi che non fanno parte dell'albero di supporto. La procedura per tale calcolo verrà illustrata sul nostro esempio. Prendiamo una qualsiasi variabile fuori base e quindi un qualsiasi arco che non faccia parte dell'albero di supporto, ad esempio l'arco $(1,3)$. Per prima cosa aggiungiamo l'arco all'albero. Si formerà esattamente un ciclo che verrà orientato nel verso dell'arco $(1,3)$ e quindi il ciclo sarà

$$1 \to 3 \to 4 \to 5 \to 1$$

come si vede da Figura 11.3. Si noti che il ciclo attraversa gli archi $(1,3)$, $(3,4)$

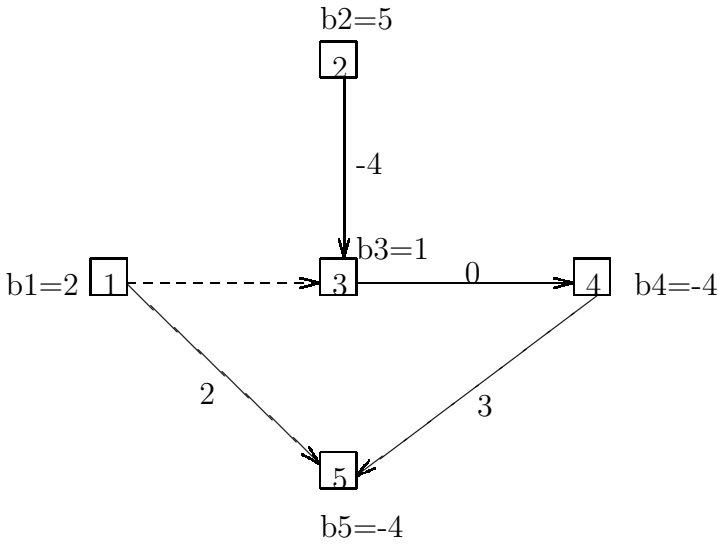


Figura 11.3: Il ciclo che si forma aggiungendo l'arco $(1,3)$.

e $(4,5)$ nel loro verso, mentre attraversa l'arco $(1,5)$ nel suo verso opposto. Il coefficiente di costo ridotto relativo all'arco $(1,3)$, indicato con $\overline{c}_{13}$ verrà calcolato sommando tra loro tutti i costi relativi agli archi attraversati dal ciclo nel loro stesso verso e sottraendo al risultato i costi degli archi attraversati dal ciclo in senso opposto al loro verso. Quindi

$$\overline{c}_{13} = c_{13} + c_{34} + c_{45} - c_{15} = -2 + 0 + 3 - 2 = -1.$$

Si noti che il coefficiente di costo ridotto è negativo e questo ci dice immediatamente che non possiamo concludere che la soluzione di base corrente è ottima. Possiamo ripetere la procedura per tutti gli archi fuori base per calcolare tutti

i coefficienti di costo ridotto. Si ottengono i seguenti risultati:

$$\overline{c}_{42} = 2 \quad \overline{c}_{12} = 2 \quad \overline{c}_{53} = 7.$$

Come già osservato, la presenza di un coefficiente di costo ridotto negativo ($\overline{c}_{13}$) ci impedisce di concludere che la soluzione di base corrente è ottima. In questi casi nel metodo del simplesso si procede a un cambio di base attraverso un'operazione di cardine. Vediamo ora come questo viene fatto nel simplesso su rete.

11.1.4 Cambio di base ovvero l'operazione di cardine nel simplesso su rete

Per il cambio di base dovremo dare una regola per stabilire quale variabile fuori base far entrare in base e quale in base dovrà uscire dalla base. Per quanto riguarda la variabile fuori base da far entrare in base, la scelta è ristretta alle sole variabili con coefficiente di costo ridotto negativo (le sole incrementando le quali si può far diminuire il costo complessivo del flusso). Tra queste fisseremo come regola di scegliere quella (o una di quelle, se sono più di una) con il coefficiente di costo ridotto il più negativo possibile. Nel nostro esempio non abbiamo alcuna scelta da fare visto che la sola variabile fuori base con coefficiente di costo ridotto negativo è quella relativa all'arco $(1, 3)$. Aggiungiamo tale arco all'albero e riotteniamo la Figura 11.3. Inizialmente il flusso lungo l'arco $(1, 3)$ è nullo ($x_{13} = 0$). Incrementiamo a Δ il valore di tale flusso. Quindi avremo un nuovo flusso pari a Δ in uscita dal nodo 1 ed in entrata al nodo 3. Per poter continuare a rispettare i vincoli relativi al nodo 1 e 3 dovremo diminuire di Δ il flusso lungo l'arco $(1, 5)$ ed aumentare di Δ il flusso lungo l'arco $(3, 4)$. A questo punto per soddisfare il vincolo relativo al nodo 4 dobbiamo incrementare di Δ il flusso lungo l'arco $(4, 5)$. Si noti che il vincolo relativo al nodo 5 è ancora soddisfatto poiché nel nodo 5 arriva un flusso pari a Δ in più dal nodo 4 ma anche un flusso ancora pari a Δ in meno dal nodo 1. Gli archi relativi al nodo 2 non subiscono variazioni e quindi il vincolo relativo al nodo 2 continua a essere soddisfatto. Si può riassumere quanto visto nel modo seguente. Una volta aggiunto l'arco $(1, 3)$ si forma un ciclo che viene orientato nel verso dell'arco $(1, 3)$ stesso. Il flusso viene incrementato di Δ lungo ogni arco che il ciclo attraversa nel suo stesso verso e decrementato di Δ lungo gli archi che vengono attraversati in verso opposto. Quindi nel nostro esempio:

$$x_{13} = \Delta \quad x_{34} = 6 + \Delta \quad x_{45} = 2 + \Delta \quad x_{15} = 2 - \Delta.$$

A questo punto possiamo incrementare il valore di Δ arrestandoci nel momento in cui un flusso lungo un arco del ciclo si annulla. Nel nostro caso possiamo incrementare Δ fino a 2 ma non oltre in quanto incrementandolo oltre il flusso relativo all'arco $(1, 5)$ diventerebbe negativo. La prima variabile che diventa nulla incrementando Δ corrisponderà alla variabile da far uscire di base. Se più variabili diventano nulle contemporaneamente incrementando Δ (caso degenere) se ne seleziona una di esse arbitrariamente. L'albero di supporto corrispondente

alla nuova base sarà quello ottenuto inserendo l'arco relativo alla variabile fatta entrare in base (l'arco $(1,3)$ nel nostro esempio) e rimuovendo l'arco della variabile fatta uscire di base (l'arco $(1,5)$ nel nostro esempio). Per il nostro esempio la nuova base è quella riportata in Figura 11.4 e i nuovi valori delle variabili sono i seguenti

$$x_{13} = 2 \;\; x_{23} = 5 \;\; x_{34} = 8 \;\; x_{45} = 4 \;\; x_{12} = x_{15} = x_{42} = x_{53} = 0$$

NOTA BENE Se il ciclo ottenuto aggiungendo all'albero di supporto l'arco

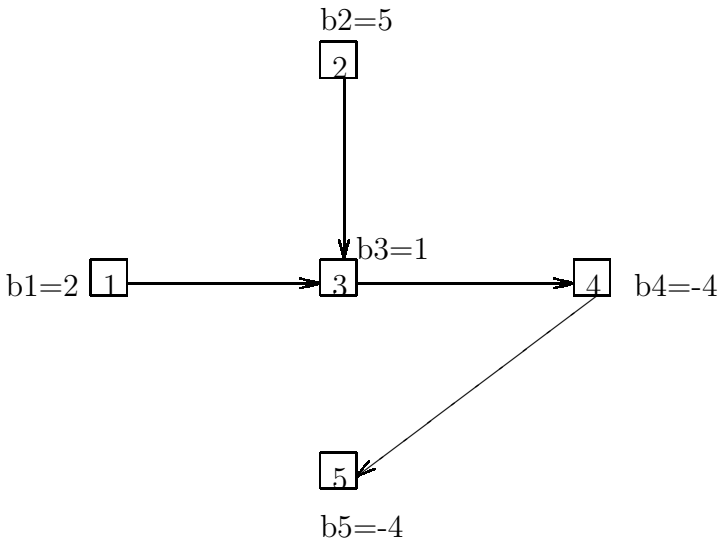


Figura 11.4: La nuova base (albero di supporto) del problema.

relativo alla variabile fuori base avesse tutti gli archi orientati nello stesso verso del ciclo stesso (vedi Figura 11.5) allora potremmo far crescere Δ all'infinito senza che nessun flusso si annulli (tutti i flussi lungo il ciclo vengono incrementati). Ciò corrisponde al caso di problema con obiettivo illimitato.

Possiamo ora concludere il nostro esempio andando a calcolare i nuovi coefficienti di costo ridotto. I risultati sono i seguenti.

$$\overline{c}_{42} = 2 \;\; \overline{c}_{15} = 1 \;\; \overline{c}_{12} = 3 \;\; \overline{c}_{53} = 7.$$

Essendo tutti non negativi si conclude che la soluzione corrente è ottima. Più precisamente, essendo tutti non solo non negativi ma anche strettamente positivi, si conclude che la soluzione è anche l'unica soluzione ottima.

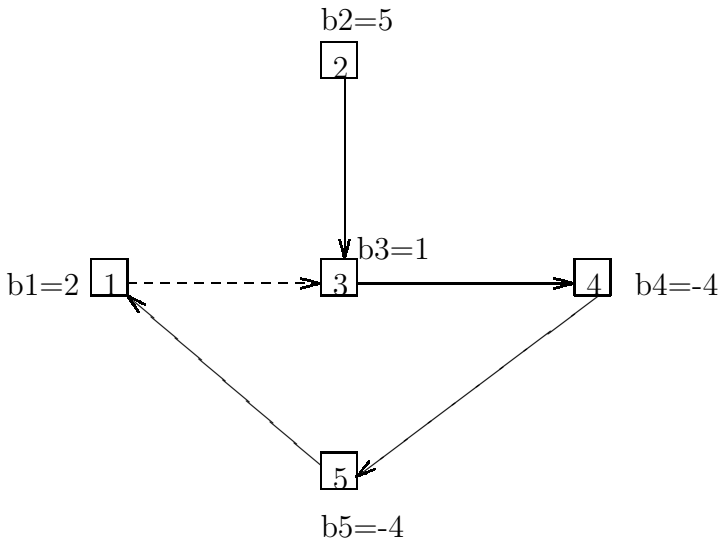


Figura 11.5: Tutti gli archi del ciclo hanno lo stesso orientamento: il problema ha obiettivo illimitato.

11.1.5 Determinare una soluzione di base ammissibile iniziale

Nella descrizione del simplesso su rete siamo partiti assumendo di avere già a disposizione un albero di supporto ammissibile. Non sempre però questo è vero e non è neppure detto che una soluzione ammissibile esista. Avremo quindi bisogno di una procedura che ci dica se ci sono soluzioni ammissibili e, nel caso esistano, ce ne restituisca una. Utilizzeremo una tecnica due fasi. Nella prima fase aggiungiamo alla nostra rete un nuovo nodo q e congiungiamo tale nodo con ogni nodo i della rete tale che $b_i < 0$ attraverso l'arco (q, i), mentre lo congiungiamo con ogni nodo i della rete tale che $b_i \geq 0$ attraverso l'arco (i, q). I valori b_i vengono lasciati invariati, mentre si pone $b_q = 0$. I costi dei flussi unitari saranno posti uguali a 1 per tutti gli archi incidenti sul nodo q e 0 per tutti gli archi della rete originaria. Per il nostro esempio la nuova rete sarà quella in Figura 11.6. Per questo problema si ha immediatamente a disposizione un albero di supporto ammissibile, quello formato da tutti gli archi incidenti su q, con i seguenti valori delle variabili:

$$x_{qi} = -b_i \quad \forall\, i:\ b_i < 0$$
$$x_{iq} = b_i \quad \forall\, i:\ b_i \geq 0$$

mentre tutte le altre variabili sono nulle. A questo punto risolviamo questo problema con il simplesso su rete nel modo già visto in precedenza. Se la soluzione ottima di tale problema è maggiore di 0, allora il problema originario ha regione ammissibile vuota. Se invece la soluzione ottima è pari a 0 e l'albero di supporto

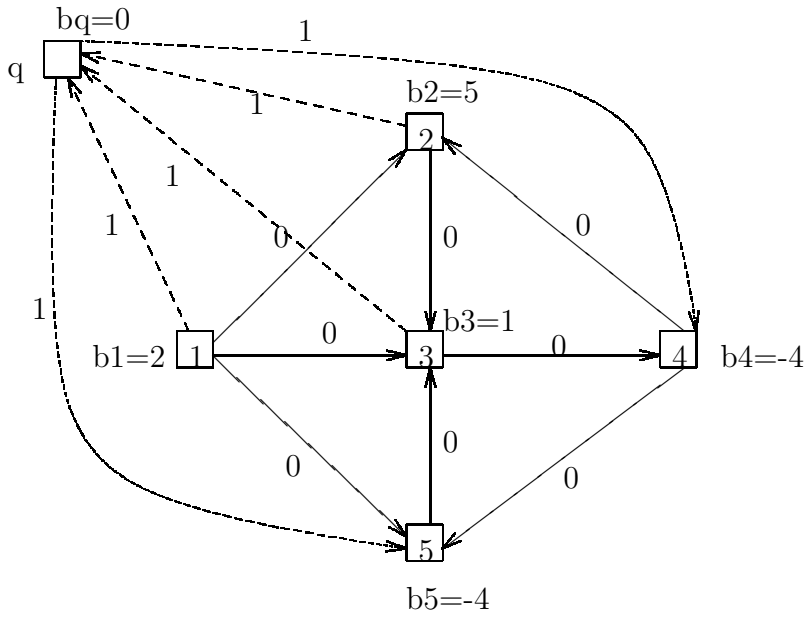


Figura 11.6: Il problema di prima fase per determinare una soluzione ammissibile iniziale.

ottimo contiene solo uno dei nuovi archi (quelli incidenti su q), eliminando tale arco si ottiene un albero di supporto ammissibile per il problema originario. A questo punto possiamo eliminare il nodo q e tutti gli archi incidenti su di esso, ripristinare gli originari costi degli archi e cominciare a risolvere il problema (seconda fase del metodo). Non illustreremo la prima fase del metodo sul nostro solito esempio in quanto ci sarebbero troppi calcoli da fare. La illustreremo su un esempio di più piccole dimensioni.

Esempio 38 *Si consideri la rete in Figura 11.7. Nella prima fase aggiungiamo il nodo q e gli archi incidenti su di esso ed aggiorniamo i costi dei flussi come indicato in Figura 11.8. Per il problema della prima fase un albero di supporto ammissibile è quello formato dagli archi incidenti su q, ovvero $(1, q)$, $(2, q)$ e $(q, 3)$. La soluzione iniziale è*

$$x_{q3} = 4 \; x_{2q} = 3 \; x_{1q} = 1,$$

tutte le altre variabili nulle. Il calcolo dei coefficienti di costo ridotto resituisce

$$\overline{c}_{12} = 0 \; \overline{c}_{13} = -2 \; \overline{c}_{23} = -2.$$

La soluzione non è ottima in quanto abbiamo coefficienti di costo ridotto negativi. Scelgo una delle variabili fuori base con coefficiente di costo ridotto più negativo, ad esempio quella associata all'arco $(1, 3)$. Applicando la procedura

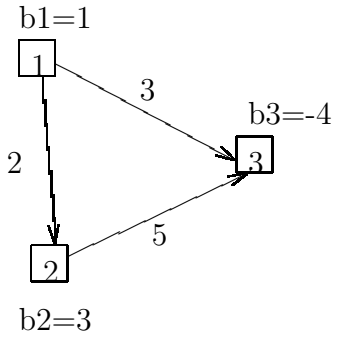


Figura 11.7: Una rete.

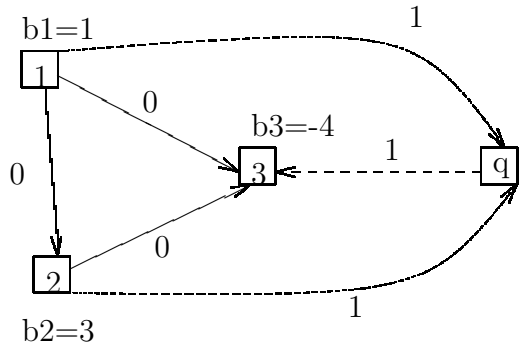


Figura 11.8: La rete ausiliaria per determinare un flusso ammissibile iniziale per la rete di Figura 11.7.

per il cambio di base ottengo il nuovo albero di supporto $(1,3)$, $(2,q)$ *e* $(q,3)$. *La nuova soluzione è*

$$x_{q3} = 3 \; x_{2q} = 3 \; x_{13} = 1,$$

tutte le altre variabili nulle. Il calcolo dei coefficienti di costo ridotto resituisce

$$\overline{c}_{12} = 2 \; \overline{c}_{1q} = 2 \; \overline{c}_{23} = -2.$$

La soluzione non è ottima in quanto abbiamo coefficienti di costo ridotto negativi. Scelgo una delle variabili fuori base con coefficiente di costo ridotto più negativo, in tal caso c'è solo quella associata all'arco $(2,3)$. *Applicando la procedura per il cambio di base ottengo il nuovo albero di supporto* $(1,3)$, $(2,3)$ *e* $(q,3)$. *La nuova soluzione è*

$$x_{q3} = 0 \; x_{23} = 3 \; x_{13} = 1,$$

tutte le altre variabili nulle. Il calcolo dei coefficienti di costo ridotto resituisce

$$\overline{c}_{12} = 0 \; \overline{c}_{1q} = 2 \; \overline{c}_{2q} = 2.$$

La soluzione è ottima ed è pari a 0. Quindi il problema ammette soluzioni ammissibili. Inoltre, poiché la soluzione ottima contiene un solo arco incidente sul nodo q, eliminando tale arco ottengo immediatamente un albero di supporto ammissibile per il problema originario (quello formato dagli archi $(1,3)$ *e* $(2,3)$*) e con tale albero di supporto ammissibile sono pronto ad entrare nella seconda fase e risolvere il problema originario.*

Una possibile semplificazione nella definizione del problema di I fase è la seguente. Si congiunga q con i soli nodi sorgente e destinazione. In tal caso la base ammissibile iniziale per il problema di I fase sarà formata da tutte gli archi incidenti su q a cui si aggiungono altri archi (incidenti su nodi transito e con flusso iniziale nullo lungo essi) fino a formare un albero di supporto.

11.2 Problemi di flusso a costo minimo con capacità limitate sugli archi

Un'importante variante dei problemi di flusso a costo minimo che abbiamo trattato sino a ora è quella in cui esistono dei limiti di capacità associati agli archi. In pratica a ogni arco $(i,j) \in A$ della rete è associato un valore intero, indicato nel seguito con d_{ij}, che rappresenta il flusso massimo di prodotto inviabile lungo quell'arco. Se si pensa all'esempio pratico di una rete di comunicazione, non si può inviare una quantità infinita di prodotto nell'unità di tempo lungo un cavo della rete, cioè si ha proprio un limite di capacità del collegamento.

Dal punto di vista della risoluzione non potremo più utilizzare l'algoritmo del simplesso come lo abbiamo visto in precedenza ma dovremo utilizzare una sua variante in grado di trattare variabili con limitazioni superiori. Resta del tutto invariata la corrispondenza uno ad uno tra basi e alberi di supporto. Ciò che cambia è che se in precedenza in corrispondenza di ogni base una variabile poteva trovarsi in uno solo tra due possibili stati (in base oppure fuori base), ora i possibili stati di una variabile x_{ij} sono tre:

1. in base;
2. fuori base *a valore nullo* (ovvero $x_{ij} = 0$);
3. fuori base *a valore pari al proprio limite superiore* (ovvero $x_{ij} = d_{ij}$).

Mentre i primi due casi coincidono con quelli già noti, la novità è rappresentata dal terzo caso. Nel seguito indicheremo con B l'insieme delle variabili in base, con N_0 quello delle variabili fuori base a valore nullo, con N_1 quello delle variabili fuori base a valore pari al proprio limite superiore. Se nel caso precedente (senza limiti di capacità) una soluzione di base associata a una base data era immediatamente identificata una volta note quali erano le variabili in base, ora per poter calcolare la soluzione di base ammissibile è necessario anche sapere per ogni variabile fuori base se essa lo sia a valore nullo oppure a valore pari

al proprio limite superiore. Una soluzione di base si definisce ammissibile se tutte le variabili in base hanno valore compreso tra 0 e la propria limitazione superiore. Inoltre, si parlerà di soluzione di base ammissibile non degenere nel caso in cui nessuna variabile in base abbia valore pari a 0 o alla propria limitazione superiore. Di seguito introdurremo un esempio e illustreremo su di esso quanto detto.

Esempio 39 *Si consideri un problema di flusso a costo minimo su una rete con 5 nodi con i seguenti valori b_i:*

$$b_1 = 8 \quad b_4 = -8 \quad b_2 = b_3 = b_5 = 0$$

e archi aventi i seguenti costi unitari di trasoprto e limiti di capacità:

	$(1,2)$	$(1,3)$	$(1,4)$	$(2,4)$	$(3,4)$	$(3,5)$	$(5,4)$
c_{ij}	5	2	15	3	2	3	2
d_{ij}	9	7	3	11	2	7	7

Il modello matematico di tale problema (una volta eliminato il vincolo relativo al nodo 5, cosa che è possibile effettuare anche in questa variante del problema) è il seguente

$$\begin{gathered}
\min \ 5x_{12} + 2x_{13} + 15x_{14} + 3x_{24} + 2x_{34} + 3x_{35} + 2x_{54} \\
x_{12} + x_{13} + x_{14} = 8 \\
x_{24} - x_{12} = 0 \\
x_{34} + x_{35} - x_{13} = 0 \\
-x_{14} - x_{24} - x_{34} - x_{54} = -8 \\
x_{12} \leq 9 \quad x_{13} \leq 7 \\
x_{14} \leq 3 \quad x_{24} \leq 11 \\
x_{34} \leq 2 \quad x_{35} \leq 7 \\
x_{54} \leq 7 \\
x_{12}, x_{13}, x_{14}, x_{24}, x_{34}, x_{35}, x_{54} \geq 0 \ \textit{interi}
\end{gathered}$$

Consideriamo ora la base formata dalle 4 variabili $\{x_{12}, x_{13}, x_{24}, x_{35}\}$ (verificate che si tratta di un albero di supporto). Si sa inoltre che delle variabili fuori base due sono fuori base a valore nullo (la x_{34} e la x_{54}), mentre una è fuori base a valore pari al proprio limite superiore (la x_{14}). Quindi abbiamo

$$B = \{x_{12}, x_{13}, x_{24}, x_{35}\} \quad N_0 = \{x_{34}, x_{54}\} \quad N_1 = \{x_{14}\}.$$

A questo punto, per ottenere il valore delle variabili in base non devo fare altro che sostituire nei vincoli a ogni variabile fuori base il valore corrispondente (0 o il limite superiore della variabile) e risolvere quindi il sistema risultante. Nel

nostro caso il sistema da risolvere sarà il seguente:

$$x_{12} + x_{13} + 3 = 8$$
$$x_{24} - x_{12} = 0$$
$$x_{35} - x_{13} = 0$$
$$-3 - x_{24} = -8$$

da cui si ottiene la soluzione di base:

$$x_{12} = 5 \quad x_{24} = 5 \quad x_{13} = 0 \quad x_{35} = 0 \quad x_{34} = x_{54} = 0 \quad x_{14} = 3$$

con valore dell'obiettivo pari a 85. Tale soluzione di base è ammissibile ed è degenere (le variabili in base x_{13} e x_{35} hanno valore nullo).

Il calcolo dei coefficienti di costo ridotto è del tutto identico a quanto già visto. In particolare per la base del nostro esempio abbiamo:

$$\overline{c}_{14} = 7 \quad \overline{c}_{34} = -4 \quad \overline{c}_{54} = -1$$

Per le variabili fuori base a valore nullo vale il discorso già fatto in precedenza: tra queste quelle che conviene far crescere se vogliamo ridurre il valore dell'obiettivo sono quelle a coefficiente di costo ridotto negativo. La novità è rappresentata dal fatto che per le variabili fuori base con valore pari al proprio limite superiore non è ovviamente possibile farle crescere (sono, appunto, già al loro limite superiore) ma solo farle decrescere. Ne consegue che per variabili fuori base al proprio limite superiore quelle la cui variazione (ovvero diminuzione) consente un decremento del valore dell'obiettivo sono quelle con coefficiente di costo ridotto positivo. Avremo quindi la seguente condizione di ottimalità: una soluzione di base ammissibile è soluzione ottima del problema se:

a) tutte le variabili fuori base a valore nullo hanno coefficiente di costo ridotto non negativo;

b) tutte le variabili fuori base a valore pari al limite superiore hanno coefficiente di costo ridotto non positivo.

Formalmente:

$$\overline{c}_{ij} \geq 0 \ \ \forall\ (i,j) \in N_0 \qquad \overline{c}_{ij} \leq 0 \ \ \forall\ (i,j) \in N_1.$$

Nel nostro esempio la condizione non è soddisfatta e si dovrà procedere a un cambio di base.

La variabile che si decide di far entrare in base è la più promettente (quella la cui variazione modifica più rapidamente il valore dell'obiettivo). Questa la si identifica prendendo quella che fornisce il massimo tra i coefficienti di costo ridotto delle variabili fuori base a valore pari al limite superiore e quelli cambiati di segno delle variabili fuori base a valore nullo. Formalmente:

$$(i,j) \in \arg\max\{\max_{(i,j)\in N_1} \overline{c}_{ij}, \max_{(i,j)\in N_0} -\overline{c}_{ij}\}$$

Nel nostro esempio si tratta di confrontare tra loro il coefficiente di costo ridotto della variabile fuori base x_{14} (pari a 7) e i coefficienti di costo ridotto cambiati di segno di x_{35} e x_{54} (rispettivamente pari a 4 e 1). Il massimo è raggiunto da x_{14} e quindi questa sarà la variabile che tenteremo di far entrare in base (in caso di parità tra più variabili se ne sceglie una a caso).

Per quanto riguarda la scelta della variabile che dovrà uscire dalla base, il procedimento è simile a quanto visto nel caso senza capacità sugli archi ma con qualche variante. Se la variabile che decidiamo di far entrare in base è fuori base a valore nullo, allora si procede come nel caso senza capacità, ovvero si fa passare da 0 a Δ il flusso lungo l'arco che si fa entrare in base e si modificano secondo le regole viste in precedenza i flussi lungo gli archi del ciclo che si viene a formare aggiungendo l'arco che si vuol far entrare in base all'albero di supporto relativo alla base attuale. Invece nel caso in cui la variabile che stiamo cercando di far entrare in base sia fuori base a valore pari al proprio limite superiore si riduce il valore di tale variabile dal proprio limite superiore d_{ij} a $d_{ij} - \Delta$ mentre sugli archi del ciclo che si forma con l'aggiunta dell'arco relativo a questa variabile i flussi sono aggiornati secondo regole inverse rispetto a quelle viste in precedenza (ovvero il flusso viene diminuito di Δ lungo gli archi attraversati secondo il proprio verso dal ciclo e incrementato di Δ lungo gli archi attarversati dal ciclo in verso opposto al proprio).
Ma fino a quanto possiamo far crescere il valore Δ? La crescita può avvenire fino a quando si verifica uno dei seguenti eventi:

1. una variabile in base si annulla (in tal caso essa uscirà dalla base e diventerà fuori base a valore nullo, cioè passerà in N_0, mentre in B entrerà la nostra variabile fuori base);

2. una variabile in base raggiunge il proprio limite superiore (in tal caso essa uscirà dalla base e diventerà fuori base a valore pari al proprio limite superiore, cioè passerà in N_1, mentre in B entrerà la nostra variabile fuori base);

3. la variabile fuori base che stiamo cercando di far entrare in base raggiunge il proprio limite superiore (se era a valore nullo) o si annulla (se era a valore pari al proprio limite superiore): in tal caso la base B non cambia, cambia solamente lo stato della variabile fuori base che abbiamo tentato di far entrare in base, la quale passa da fuori base a valore nullo a fuori base a valore pari al proprio limite superiore (cioè da N_0 in N_1) o viceversa.

Ma vediamo di capire cosa succede nel nostro esempio. Se cerchiamo di far entrare in base x_{14} abbiamo le seguenti variazioni nei valori delle variabili

$$x_{12} = 5 + \Delta \quad x_{24} = 5 + \Delta \quad x_{13} = 0 \quad x_{35} = 0 \quad x_{34} = x_{54} = 0 \quad x_{14} = 3 - \Delta$$

con un valore dell'obiettivo pari a $85 - 7\Delta$. Si vede che Δ può crescere al massimo fino al valore 3 (in corrispondenza di tale valore la variabile x_{14} si annulla). Quindi la base non cambia ma cambia lo stato della variabile x_{14} che

da fuori base a valore pari al proprio limite superiore passa a fuori base a valore nullo. La nuova soluzione di base è la seguente:

$$x_{12} = 8 \quad x_{24} = 8 \quad x_{13} = 0 \quad x_{35} = 0 \quad x_{34} = x_{54} = x_{14} = 0$$

con un valore dell'obiettivo pari a 64. In questo momento abbiamo:

$$B = \{x_{12}, x_{13}, x_{24}, x_{35}\} \quad N_0 = \{x_{14}, x_{34}, x_{54}\}.$$

A questo punto la procedura viene iterata. Calcoliamo i coefficienti di costo ridotto (in realtà non essendo cambiata la base essi rimangono identici a prima):

$$\overline{c}_{14} = 7 \quad \overline{c}_{34} = -4 \quad \overline{c}_{54} = -1.$$

La condizione di ottimalità non è soddisfatta (i coefficienti di costo ridotto di x_{34} e x_{54}, entrambe in N_0, sono negativi). La variabile che si decide di far entrare in base è la x_{34}. Se cerchiamo di far entrare in base x_{34} abbiamo le seguenti variazioni nei valori delle variabili

$$x_{12} = 8 - \Delta \quad x_{24} = 8 - \Delta \quad x_{13} = \Delta \quad x_{35} = 0 \quad x_{34} = \Delta \quad x_{54} = x_{14} = 0$$

con un valore dell'obiettivo pari a $64 - 4\Delta$. Si vede che Δ può crescere al massimo fino al valore 2 (in corrispondenza di tale valore la variabile x_{34} raggiunge il proprio limite superiore). Quindi la base non cambia ma cambia lo stato della variabile x_{34} che da fuori base a valore nullo passa a fuori base a valore pari al proprio limite superiore. La nuova soluzione di base è la seguente:

$$x_{12} = 6 \quad x_{24} = 6 \quad x_{13} = 2 \quad x_{35} = 0 \quad x_{34} = 2 \quad x_{54} = x_{14} = 0$$

con un valore dell'obiettivo pari a 56. In questo momento abbiamo:

$$B = \{x_{12}, x_{13}, x_{24}, x_{35}\} \quad N_0 = \{x_{14}, x_{54}\} \quad N_1 = \{x_{34}\}.$$

Continuiamo con la nostra procedura. Calcoliamo i coefficienti di costo ridotto (in realtà non essendo cambiata la base essi continuano a rimanere identici a prima):

$$\overline{c}_{14} = 7 \quad \overline{c}_{34} = -4 \quad \overline{c}_{54} = -1.$$

La condizione di ottimalità non è soddisfatta (il coefficiente di costo ridotto di x_{54}, che è in N_0, è negativo). La variabile che si decide di far entrare in base è la x_{54}. Se cerchiamo di far entrare in base x_{54} abbiamo le seguenti variazioni nei valori delle variabili

$$x_{12} = 6 - \Delta \quad x_{24} = 6 - \Delta \quad x_{13} = 2 + \Delta \quad x_{35} = \Delta \quad x_{54} = \Delta \quad x_{34} = 2 \quad x_{14} = 0$$

con un valore dell'obiettivo pari a $56 - \Delta$. Si vede che Δ può crescere al massimo fino al valore 5 (in corrispondenza di tale valore la variabile in base x_{13} raggiunge il proprio limite superiore). Quindi la base ora cambia in quanto in essa la variabile x_{13} viene sostituita dalla x_{54}. La nuova soluzione di base è la seguente:

$$x_{12} = 1 \quad x_{54} = 5 \quad x_{13} = 7 \quad x_{35} = 5 \quad x_{34} = 2 \quad x_{24} = 1 \quad x_{14} = 0$$

con un valore dell'obiettivo pari a 51. In questo momento abbiamo:

$$B = \{x_{12}, x_{24}, x_{54}, x_{35}\} \quad N_0 = \{x_{14}\} \quad N_1 = \{x_{13}, x_{34}\}.$$

Calcoliamo i coefficienti di costo ridotto:

$$\overline{c}_{14} = 7 \quad \overline{c}_{34} = -3 \quad \overline{c}_{13} = -1.$$

Da essi possiamo concludere che la soluzione di base ammissibile attuale è soluzione ottima (unica) del nostro problema.

Il problema di stabilire se esistono o meno soluzioni ammissibili del problema si risolve con una procedura a due fasi del tutto analoga a quella vista per il problema senza vincoli di capacità sugli archi (nel problema di I fase agli archi incidenti sul nodo aggiuntivo q si assegna capacità infinita, mentre quelli originari mantengono la loro capacità).

11.3 Flusso massimo

Prima di addentrarci nella descrizione dell'algoritmo di Ford-Fulkerson per il problema di flusso massimo, dobbiamo introdurre un problema strettamente legato a quello di flusso massimo, il problema di taglio minimo sulla stessa rete. L'algoritmo proposto fornirà contemporaneamente anche una soluzione per tale problema.

11.3.1 Tagli e problema del taglio minimo

Si consideri $U \subset V$ con la proprietà che:

$$S \in U \quad D \notin U.$$

L'insieme di archi

$$\mathcal{S}_U = \{(i,j) \in A \ : \ i \in U, \ j \notin U\},$$

ovvero gli archi con il primo estremo in U e l'altro al di fuori di U, viene detto *taglio* della rete. Si noti che eliminando tutti gli archi di un taglio dalla rete rendo impossibile raggiungere D a partire da S in quanto ciò equivale ad eliminare tutti gli archi che vanno da U (contenente S) al suo complementare $\overline{U} = V \setminus U$ (contenente D). Ad un taglio si associa un *costo* pari alla somma delle capacità degli archi del taglio, cioè:

$$C(\mathcal{S}_U) = \sum_{(i,j) \in \mathcal{S}_U} c_{ij}.$$

Se pensiamo alla rete descritta nell'Esempio 8, l'insieme $U = \{S, n_1, n_2\}$ induce il taglio $\mathcal{S}_U = \{(n_1, n_3), (n_1, n_4), (n_2, n_3), (n_2, n_4)\}$ con capacità $C(\mathcal{S}_U) = 7$. Si può notare che, dato un taglio $\mathcal{S}_U$ qualsiasi, il valore del flusso massimo nella rete non può superare quello del costo del taglio. Infatti, per poter passare dall'insieme di nodi U contenente la sorgente S al suo complementare $\overline{U}$ contenente la destinazione D il flusso può solo passare attraverso gli archi del taglio $\mathcal{S}_U$ e quindi il flusso non può superare la somma delle capacità di tali archi, ovvero il costo del taglio. Quindi il costo di ogni taglio rappresenta un limite superiore per il valore del flusso massimo (nel nostro esempio sappiamo quindi già che il flusso massimo non può superare il valore 7, ovvero il costo del taglio indotto da $U = \{S, n_1, n_2\}$). Possiamo anche spingerci più in là e dire che tra tutti i tagli ve ne è almeno uno il cui costo è *esattamente pari* a quello del flusso massimo. Più precisamente, consideriamo il problema del taglio di costo minimo che consiste nel determinare, tra tutti i tagli possibili all'interno di una rete, quello il cui costo sia il più piccolo possibile, ovvero

$$\min_{U \subset V \ : \ S \in U, \ D \notin U} C(\mathcal{S}_U). \tag{11.2}$$

Avremo modo di dimostrare che i valori ottimi del problema di flusso massimo e quello di taglio a costo minimo sono uguali tra loro. Non solo, l'algoritmo con

cui risolveremo il problema di flusso massimo ci darà immediatamente anche una soluzione per il problema di taglio a costo minimo. Richiamando le nozioni di dualità nella PL, possiamo dire che il problema di taglio a costo minimo è un problema *duale* del problema di flusso massimo.

11.3.2 Algoritmo di Ford-Fulkerson

Ci occuperemo ora di descrivere un possibile algoritmo di risoluzione per il problema di flusso massimo, chiamato *algoritmo di Ford-Fulkerson*. Prima di descrivere l'algoritmo abbiamo bisogno di introdurre alcuni concetti. Supponiamo di avere un flusso ammissibile:

$$\overline{X} = (\overline{x}_{ij})_{(i,j)\in A},$$

ovvero un flusso che soddisfa i vincoli di equilibrio e quelli di capacità degli archi. Se $\overline{x}_{ij} = c_{ij}$ diremo che l'arco (i, j) è *saturo*. Si consideri ora un cammino orientato nella rete dal nodo S al nodo D:

$$S = q_0 \to q_1 \to \cdots \to q_r \to q_{r+1} = D,$$

privo di archi saturi, ovvero nesuno degli archi

$$(q_i, q_{i+1}) \in A \quad i = 0, \ldots, r$$

è saturo. In tal caso il flusso $\overline{X}$ non è ottimo. Infatti, posso aumentare il flusso lungo ciascun arco del cammino di una quantità Δ definita nel seguente modo:

$$\Delta = \min_{i=0,\ldots,r} [c_{q_i q_{i+1}} - \overline{x}_{q_i q_{i+1}}],$$

senza violare i vincoli di capacità degli archi. Si noti anche che i vincoli di equilibrio continuano ad essere rispettati in quanto in ogni nodo intermedio del grafo facente parte del cammino si ha che il flusso in entrata aumenta di Δ ma contemporaneamente aumenta di Δ anche quello in uscita. Essendo gli archi del cammino non saturi si ha che $\Delta > 0$ e il flusso totale da S a D viene aumentato proprio di questa quantità. Per illustrare questa situazione si consideri l'esempio in Figura 11.9 dove di fianco agli archi sono riportati due numeri, il primo corrispondente al flusso attuale $\overline{x}_{ij}$ lungo l'arco, l'altro corrispondente alla capacità c_{ij} dell'arco. Attuamente il flusso in uscita dal nodo S è pari a 2 (1 unità lungo l'arco (S, n_1) e 1 lungo l'arco (S, n_2)). Questo flusso non è ottimo. Notiamo infatti che esiste almeno un cammino orientato da S a D con archi non saturi, ad esempio il cammino

$$S \to n_1 \to n_2 \to D$$

Lungo tale cammino si ha

$$\Delta = \min\{c_{Sn_1} - \overline{x}_{Sn_1},\ c_{n_1n_2} - \overline{x}_{n_1n_2},\ c_{n_2D} - \overline{x}_{n_2D}\} = 1.$$

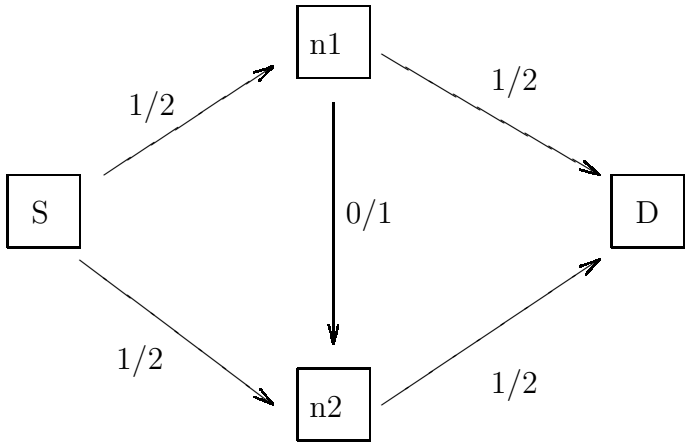


Figura 11.9: I numeri a fianco di ogni arco rappresentano rispettivamente il flusso lungo di esso e la sua capacità.

Quindi posso incrementare di 1 unità il flusso lungo gli archi del cammino, cioè avrò il seguente aggiornamento del flusso:

$$\overline{x}_{Sn_1} = \overline{x}_{Sn_1} + \Delta \quad \overline{x}_{Sn_2} = \overline{x}_{Sn_2} \quad \overline{x}_{n_1n_2} = \overline{x}_{n_1n_2} + \Delta$$

$$\overline{x}_{n_1D} = \overline{x}_{n_1D} \quad \overline{x}_{n_2D} = \overline{x}_{n_2D} + \Delta$$

La nuova situazione è illustrata in Figura 11.10. Ora il flusso totale uscente

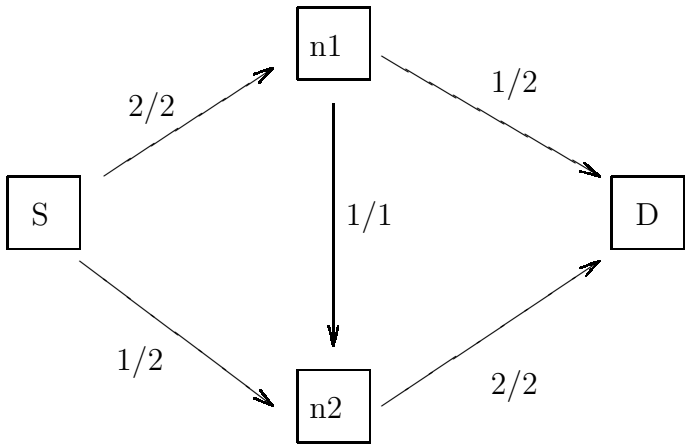


Figura 11.10: La nuova situazione dopo l'incremento del flusso.

da S è salito anch'esso della quantità Δ passando da 2 a 3. Possiamo notare che ora tutti i cammini orientati da S a D contengono almeno un arco saturo. Possiamo allora concludere che il flusso attuale è il massimo possibile? La

risposta è no e lo dimostreremo nel seguito. Prima definiamo un nuovo grafo orientato $G(\overline{X}) = (V, A(\overline{X}))$ detto *grafo associato al flusso* $\overline{X}$. Il nuovo grafo ha gli stessi nodi della rete originaria e ha il seguente insieme di archi:

$$A(\overline{X}) = \underbrace{\{(i,j) \ : \ (i,j) \in A, \ \overline{x}_{ij} < c_{ij}\}}_{A_f(\overline{X})} \cup \underbrace{\{(i,j) \ : \ (j,i) \in A, \ \overline{x}_{ji} > 0\}}_{A_b(\overline{X})},$$

ovvero $A(\overline{X})$ contiene tutti gli archi di A non saturi (l'insieme $A_f(\overline{X})$, detto insieme degli archi *forward*) e tutti gli archi di A lungo cui è stata inviata una quantità positiva di flusso, cambiati però di verso (l'insieme $A_b(\overline{X})$, detto insieme degli archi *backward*). Vediamo di generare il grafo associato al flusso attuale $\overline{X}$ del nostro esempio. Si ha che:

$$A_f(\overline{X}) = \{(S, n_2), \ (n_1, D)\}$$

$$A_b(\overline{X}) = \{(n_1, S), \ (n_2, S), \ (n_2, n_1), \ (D, n_1), \ (D, n_2)\}.$$

Il grafo è rappresentato in Figura 11.11. Poniamoci ora la seguente domanda:

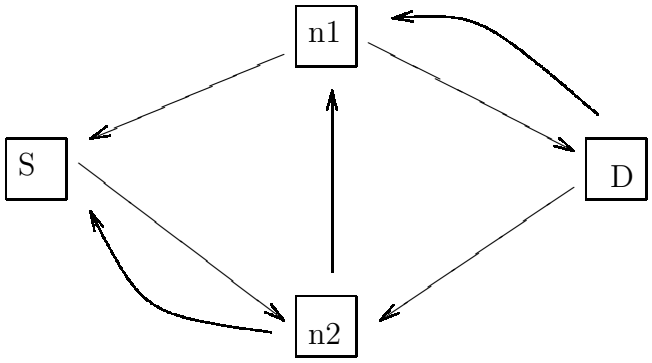


Figura 11.11: Il grafo associato al flusso corrente.

esiste sul nuovo grafo orientato $G(\overline{X})$ un cammino orientato da S a D? La risposta è affermativa. Esiste infatti il cammino orientato:

$$S \to n_2 \to n_1 \to D.$$

Indichiamo con $P = \{(S, n_2), \ (n_2, n_1), \ (n_1, D)\}$ l'insieme degli archi di tale cammino. Possiamo a questo punto modificare il nostro flusso nel modo seguente. Per ogni arco (i, j) del cammino P si calcoli il seguente valore:

$$\alpha_{ij} = \begin{cases} c_{ij} - \overline{x}_{ij} & \text{se } (i,j) \in A_f(\overline{X}) \cap P \\ \overline{x}_{ji} & \text{se } (i,j) \in A_b(\overline{X}) \cap P \end{cases}$$

e quindi sia

$$\Delta = \min_{(i,j) \in P} \alpha_{ij}$$

il minimo di tali valori. Per gli archi forward il valore α_{ij} rappresenta quanto flusso posso ancora inviare lungo l'arco $(i,j) \in A$ della rete originaria, per gli archi backward il valore α_{ij} rappresenta quanto flusso posso rispedire indietro lungo l'arco $(j,i) \in A$ della rete originaria. Nel nostro caso si ha:

$$(S,n_2) \in A_f(\overline{X}) \rightarrow \ \alpha_{Sn_2} = 2-1 = 1 \quad (n_2,n_1) \in A_b(\overline{X}) \rightarrow \ \alpha_{n_2n_1} = 1$$

$$(n_1,D) \in A_f(\overline{X}) \rightarrow \ \alpha_{n_1D} = 2-1 = 1$$

Il minimo tra questi tre valori è $\Delta = 1$. Ora il flusso viene aggiornato nel modo seguente:

$$\overline{x}_{ij} = \begin{cases} \overline{x}_{ij} + \Delta & \text{se } (i,j) \in A_f(\overline{X}) \cap P \\ \overline{x}_{ij} - \Delta & \text{se } (j,i) \in A_b(\overline{X}) \cap P \\ \overline{x}_{ij} & \text{altrimenti} \end{cases} \tag{11.3}$$

Quindi nel nostro esempio avremo:

$$\overline{x}_{Sn_1} = 2 \quad \overline{x}_{Sn_2} = 1+1 = 2 \quad \overline{x}_{n_1n_2} = 1-1 = 0$$

$$\overline{x}_{n_1D} = 1+1 = 2 \quad \overline{x}_{n_2D} = 2.$$

La nuova situazione è illustrata in Figura 11.12. In pratica sulla rete originaria

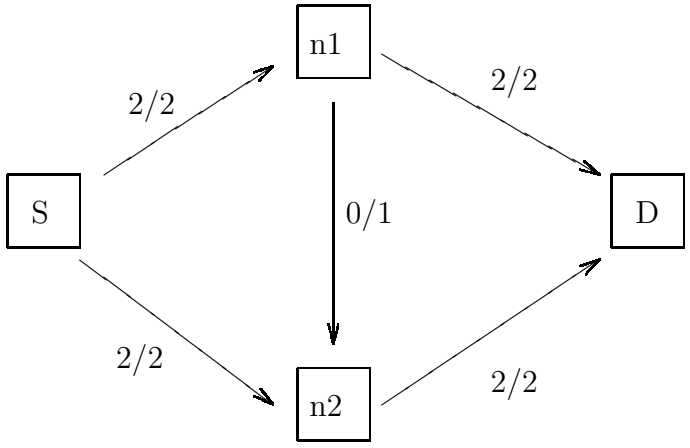


Figura 11.12: La nuova situazione dopo l'aggiornamento del flusso.

incrementiamo di Δ il flusso lungo gli archi corrispondenti ad archi forward del grafo associato al flusso attuale appartenenti al cammino P, rispediamo indietro Δ unità di flusso lungo gli archi che, cambiati di verso, corrispondono ad archi backward del grafo associato appartenenti al cammino P, e quindi decrementiamo il flusso della stessa quantità lungo tali archi. Infine, lungo gli archi che non fanno parte del cammino P il flusso rimane invariato. Si può verificare che il nuovo flusso è ammissibile. In particolare, per come è stato scelto il valore Δ il nuovo flusso lungo ciascun arco è non negativo e non supera

la capacità dell'arco. Inoltre, continuano anche ad essere soddisfatti i vincoli di equilibrio nei nodi intermedi. La quantità di flusso uscente da S è anch'essa aumentata proprio della quantità Δ e quindi passa da 3 a 4, il che dimostra che il flusso precedente non era ottimo. Con il nuovo flusso possiamo ripetere quanto visto. Costruiamo il grafo $G(\overline{X})$ associato al nuovo flusso $\overline{X}$. Ora avremo:

$$A_f(\overline{X}) = \{(n_1, n_2)\} \quad A_b(\overline{X}) = \{(n_1, S),\ (n_2, S),\ (D, n_1),\ (D, n_2)\}.$$

Il grafo è illustrato in Figura 11.13. Possiamo ora chiederci se il nuovo flus-

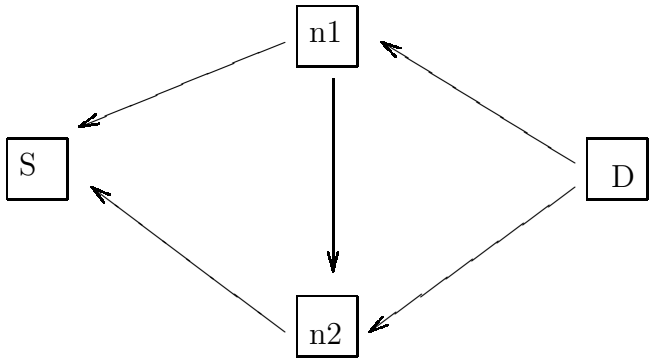


Figura 11.13: Il grafo associato al nuovo flusso.

so è quello ottimo. La risposta la possiamo dare sulla base della seguente fondamentale osservazione.

Osservazione 42 *Dato un flusso $\overline{X}$, esso è ottimo se e solo se nel grafo associato $G(\overline{X})$ non esiste alcun cammino orientato da S a D.*

Dalla Figura 11.13 si può notare che per il grafo associato al nostro nuovo flusso non esiste alcun cammino orientato dal nodo S al nodo D e quindi il flusso attuale è ottimo. Quanto visto rappresenta la base dell'algoritmo di Ford-Fulkerson. In pratica, ad ogni iterazione di tale algoritmo si eseguono i seguenti passi.

Passo 1 Si costruisca il grafo $G(\overline{X})$ associato al flusso attuale $\overline{X}$.

Passo 2 Se non esiste alcun cammino orientato da S a D in $G(\overline{X})$, allora STOP: il flusso attuale $\overline{X}$ è ottimo. Altrimenti, si individui un tale cammino P e si aggiorni il flusso $\overline{X}$ come indicato in (11.3) e si ritorni al Passo 1.

Se quella sopra è la struttura dell'algoritmo, vediamo ora di entrare nei suoi dettagli.

ALGORITMO DI FORD-FULKERSON

Passo 0 Si parta con un flusso ammissibile $\overline{X}$. Si noti che è sempre possibile partire con il flusso nullo $\overline{X} = 0$.

Passo 1 Si associ alla sorgente S l'etichetta (S, ∞). L'insieme R dei nodi *analizzati* è vuoto, ovvero $R = \emptyset$, mentre l'insieme E dei nodi *etichettati* contiene il solo nodo S, ovvero $E = \{S\}$.

Passo 2 Si verifichi se $E \setminus R \neq \emptyset$, ovvero se vi sono nodi etichettati non ancora analizzati. Se non ne esistono il flusso attuale $\overline{X}$ è ottimo. Altrimenti si selezioni un nodo $i \in E \setminus R$, cioè etichettato, con etichetta (k, Δ), ma non ancora analizzato e lo si analizzi. Analizzare il nodo i vuol dire compiere le seguenti operazioni. Per ogni nodo $j \notin E$, cioè non ancora etichettato, e tale che $(i, j) \in A(\overline{X})$, si etichetti j con la seguente etichetta:

$$(i, \min(\Delta, c_{ij} - \overline{x}_{ij})) \quad \text{se } (i, j) \in A_f(\overline{X})$$

$$(i, \min(\Delta, \overline{x}_{ji})) \quad \text{se } (i, j) \in A_b(\overline{X})$$

Quindi si ponga:

$$E = E \cup \{j \notin E \ : \ (i, j) \in A(\overline{X})\} \qquad R = R \cup \{i\}.$$

Se $D \in E$, cioè se la destinazione è stata etichettata, si vada al Passo 3, altrimenti si ripeta il Passo 2.

Passo 3 Si ricostruisca un cammino orientato da S a D in $G(\overline{X})$ procedendo a ritroso da D verso S ed usando le prime componenti delle etichette. Più precisamente, si cominci col considerare l'etichetta (q_r, Δ) di D. Allora nel cammino orientato da S a D il nodo D è preceduto da q_r. Per conoscere da quale nodo è preceduto q_r se ne consideri la prima componente dell'etichetta, indicata qui con q_{r-1}. Si ripeta quindi la stessa procedura con q_{r-1}. Ci si arresta non appena si arriva ad un nodo q_1 la cui prima componente dell'etichetta è il nodo sorgente S. A questo punto il cammino

$$S \to q_1 \to \cdots \to q_{r-1} \to q_r \to D$$

con insieme di archi

$$P = \{(S, q_1), \ldots, (q_{r-1}, q_r), (q_r, D)\}$$

è un cammino orientato da S a D in $G(\overline{X})$. Ora possiamo aggiornare $\overline{X}$ come indicato in (11.3), dove il valore Δ è dato dalla seconda componente dell'etichetta del nodo D, e ritornare al Passo 1.

Quando l'algoritmo termina abbiamo una soluzione sia per il problema di flusso massimo che per il problema di taglio minimo, come dimostra il seguente teormema.

Teorema 9 *Quando l'algoritmo termina abbiamo che il flusso ammissibile attuale è soluzione ottima del problema di flusso massimo e se si pone $U = E$, dove E è l'insieme dei nodi etichettati al momento della terminazione dell'algoritmo, si ha anche che il taglio $\mathcal{S}_U$ indotto da U è soluzione ottima del problema di taglio minimo.*

Dimostrazione Per prima cosa si noti che al momento della terminazione dell'algoritmo si ha $S \in E$ (il nodo S viene sempre etichettato al Passo 1) e $D \notin E$ (altrimenti dovremmo andare al Passo 3 ed aggiornare il flusso attuale). Quindi l'insieme E induce effettivamente un taglio. Vediamo ora qual è il valore di questo taglio. Se riusciamo a dimostrare che esso coincide con il valore del flusso uscente da S, avendo già osservato che il costo di ogni taglio è non inferiore al valore del flusso massimo, possiamo concludere che esso è il taglio a costo minimo. Per prima cosa valutiamo il flusso uscente da S ed entrante in D. Esso coincide con tutto il flusso che dai nodi in E viene spostato verso i nodi nel complemento $\overline{E} = V \setminus E$ di E meno il flusso che va in senso opposto, cioè quello dai nodi nel complemento $\overline{E}$ verso i nodi in E (si veda la Figura 11.14). In

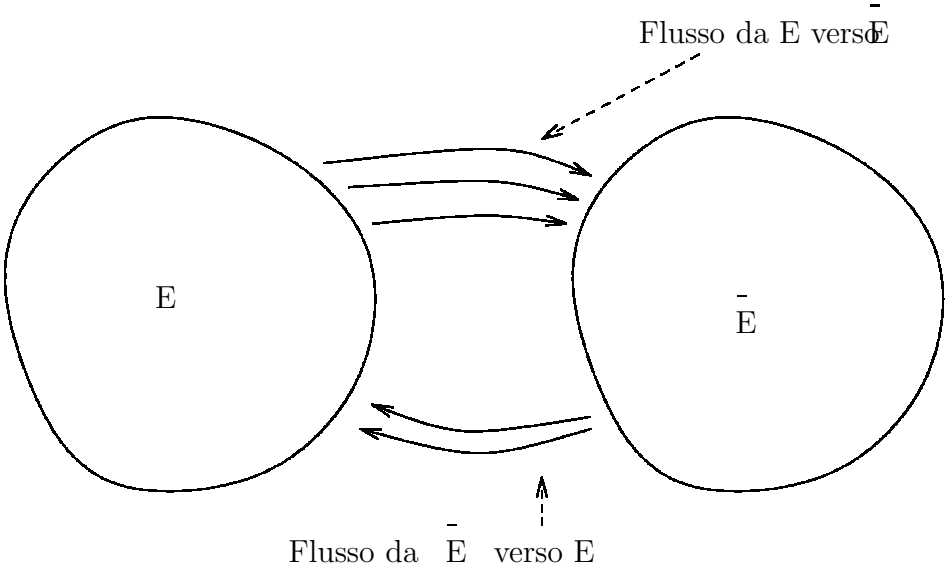


Figura 11.14: Il flusso totale dalla sorgente alla destinazione è pari alla differenza tra il flusso totale da E verso $\overline{E}$ e il flusso totale da $\overline{E}$ verso E.

formule il flusso uscente da S ed entrante in D è quindi pari a:

$$\sum_{(i,j)\ :\ (i,j)\in A,\ i\in E,\ j\in\overline{E}} \overline{x}_{ij} - \sum_{(j,i)\ :\ (j,i)\in A,\ i\in E,\ j\in\overline{E}} \overline{x}_{ji}. \qquad (11.4)$$

Ma vediamo ora come devono essere i valori $\overline{x}_{ij}$ per $(i,j) \in A,\ i \in E,\ j \in \overline{E}$ e i valori $\overline{x}_{ji}$ per $(j,i) \in A,\ i \in E,\ j \in \overline{E}$. Si deve avere che

$$\forall\ (i,j) \in A,\ i \in E,\ j \in \overline{E} \quad \overline{x}_{ij} = c_{ij}. \qquad (11.5)$$

Infatti, per assurdo si supponga che esista un $(i_1, j_1) \in A,\ i_1 \in E,\ j_1 \in \overline{E}$ con $\overline{x}_{i_1 j_1} < c_{i_1 j_1}$. In tal caso $(i_1, j_1) \in A_f(\overline{X})$ e quindi al Passo 2 dovremmo assegnare un'etichetta a j_1, il che contraddice l'ipotesi che $j_1 \notin E$. Analogamente, si deve avere che

$$\forall\ (j,i) \in A,\ i \in E,\ j \in \overline{E} \quad \overline{x}_{ji} = 0. \tag{11.6}$$

Infatti, per assurdo si supponga che esista un $(j_1, i_1) \in A,\ i_1 \in E,\ j_1 \in \overline{E}$ con $\overline{x}_{j_1 i_1} > 0$. In tal caso $(i_1, j_1) \in A_b(\overline{X})$ e quindi al Passo 2 dovremmo assegnare un'etichetta a j_1, il che contraddice ancora l'ipotesi che $j_1 \notin E$. Sostituendo (11.5) e (11.6) in (11.4) si ottiene che il valore del flusso è pari a

$$\sum_{(i,j)\ :\ (i,j)\in A,\ i\in E,\ j\in\overline{E}} c_{ij} = C(\mathcal{S}_E),$$

cioè è pari al costo del taglio indotto da E, il che conclude la dimostrazione.

Dobbiamo ancora precisare la complessità dell'algoritmo. Se si considera la rete $G = (V, A)$ con

$$V = \{S, 1, 2, D\} \quad A = \{(S,1), (S,2), (1,2), (1,D), (2,D)\}$$

e capacità degli archi:

Archi	$(S,1)$	$(S,2)$	$(1,2)$	$(1,D)$	$(2,D)$
Capacità	M	M	1	M	M

posso scegliere come percorso aumentante alternativamente $S \to 1 \to 2 \to D$ e $S \to 2 \to 1 \to D$ e in tal caso incremento di una sola unità il flusso ad ogni iterazione. Il numero di iterazioni è quindi pari a $2M$, esponenziale rispetto alla dimensione dell'istanza (perché?). Si può però dimostrare il seguente risultato.

Osservazione 43 *Se se la scelta del nodi $i \in E \setminus R$ da analizzare viene fatta secondo una disciplina FIFO, cioè i nodi vengono analizzati nello stesso ordine in cui vengono etichettati, allora il numero di operazioni richieste dall'algoritmo è pari a $O(\mid A \mid\mid V \mid^2)$.*

L'algoritmo ha quindi complessità polinomiale. Va anche sottolineato come questa non sia la migliore complessità possibile per questo problema. Esistono algoritmi più sofisticati con una complessità pari a $O(\mid V \mid^3)$.

Un esempio di applicazione dell'algoritmo

Vediamo ora di risolvere il problema di flusso massimo dell'Esempio 8. utilizzando l'algoritmo di Ford-Fulkerson. Cominciamo con il flusso nullo $\overline{X}$. In tal caso si ha che il grafo associato a tale flusso nullo coincide con il grafo originario, ovvero $G(\overline{X}) \equiv G$. Il primo ciclo dell'algoritmo è descritto nella Tabella 11.1. Il valore Δ è pari a 1 ed il cammino orientato in $G(\overline{X})$ da S a D è il seguente:

$$S \to n_1 \to n_3 \to D.$$

Tabella 11.1:

	E	R	S	n_1	n_2	n_3	n_4	D
Passo 1	S	$\emptyset$	(S,∞)	-	-	-	-	-
Passo 2	S, n_1, n_2	S	(S,∞)	$(S,3)$	$(S,2)$	-	-	-
Passo 2	S, n_1, n_2, n_3, n_4	S, n_1	(S,∞)	$(S,3)$	$(S,2)$	$(n_1,1)$	$(n_1,3)$	-
Passo 2	S, n_1, n_2, n_3, n_4	S, n_1, n_2	(S,∞)	$(S,3)$	$(S,2)$	$(n_1,1)$	$(n_1,3)$	-
Passo 2	S, n_1, n_2, n_3, n_4, D	S, n_1, n_2, n_3	(S,∞)	$(S,3)$	$(S,2)$	$(n_1,1)$	$(n_1,3)$	$(n_3,1)$

Ciò porta al seguente aggiornamento del flusso:

$$\overline{x}_{Sn_1} = 1 \;\; \overline{x}_{Sn_2} = 0 \;\; \overline{x}_{n_1n_3} = 1 \;\; \overline{x}_{n_1n_4} = 0$$

$$\overline{x}_{n_2n_3} = 0 \;\; \overline{x}_{n_2n_4} = 0 \;\; \overline{x}_{n_3D} = 1 \;\; \overline{x}_{n_4D} = 0$$

Tabella 11.2:

	E	R	S	n_1	n_2	n_3	n_4	D
Passo 1	S	$\emptyset$	(S,∞)	-	-	-	-	-
Passo 2	S, n_1, n_2	S	(S,∞)	$(S,2)$	$(S,2)$	-	-	-
Passo 2	S, n_1, n_2, n_4	S, n_1	(S,∞)	$(S,2)$	$(S,2)$	-	$(n_1,2)$	-
Passo 2	S, n_1, n_2, n_3, n_4	S, n_1, n_2	(S,∞)	$(S,2)$	$(S,2)$	$(n_2,1)$	$(n_1,2)$	-
Passo 2	S, n_1, n_2, n_3, n_4, D	S, n_1, n_2, n_4	(S,∞)	$(S,2)$	$(S,2)$	$(n_2,1)$	$(n_1,2)$	$(n_4,2)$

Il grafo $G(\overline{X})$ associato al nuovo flusso è illustrato in Figura 11.15. A questo

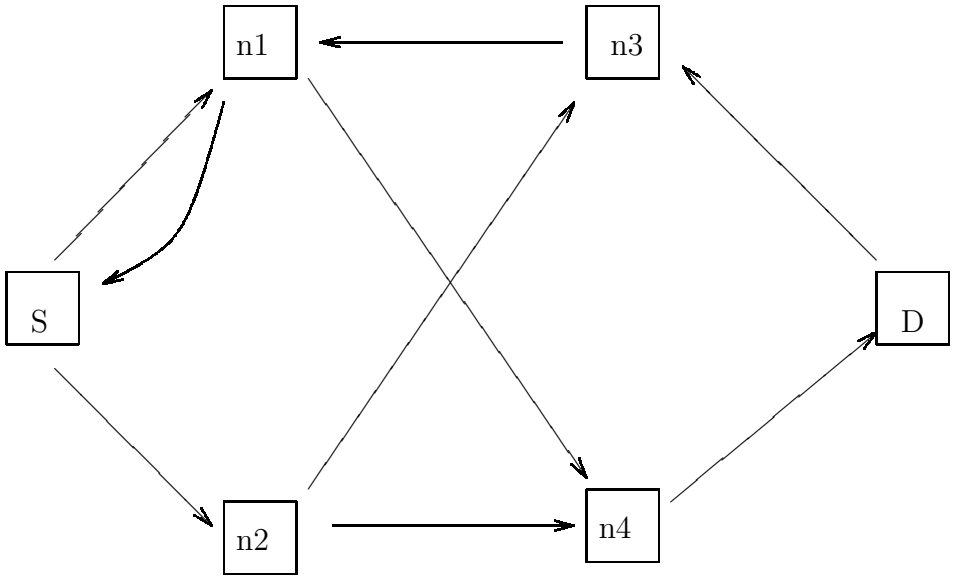


Figura 11.15: Il grafo associato al nuovo flusso.

punto si ripete la procedura con i passi indicati in Tabella 11.2. Il valore Δ è pari a 2 ed il cammino orientato in $G(\overline{X})$ da S a D è il seguente:

$$S \to n_1 \to n_4 \to D.$$

Ciò porta al seguente aggiornamento del flusso:

$$\overline{x}_{Sn_1} = 3 \;\; \overline{x}_{Sn_2} = 0 \;\; \overline{x}_{n_1n_3} = 1 \;\; \overline{x}_{n_1n_4} = 2$$

$$\overline{x}_{n_2n_3} = 0 \;\; \overline{x}_{n_2n_4} = 0 \;\; \overline{x}_{n_3D} = 1 \;\; \overline{x}_{n_4D} = 2$$

Il grafo $G(\overline{X})$ associato al nuovo flusso è illustrato in Figura 11.16. A questo

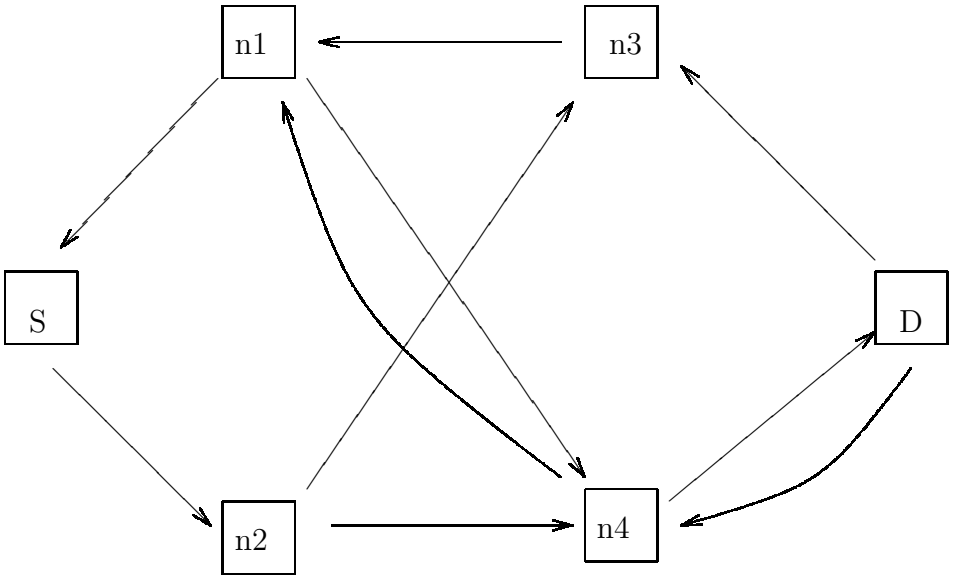


Figura 11.16:

Tabella 11.3:

	E	R	S	n_1	n_2	n_3	n_4	D
Passo 1	S	$\emptyset$	(S,∞)	-	-	-	-	-
Passo 2	S, n_2	S	(S,∞)	-	$(S,2)$	-	-	-
Passo 2	S, n_2, n_3, n_4	S, n_2	(S,∞)	-	$(S,2)$	$(n_2,1)$	$(n_2,1)$	-
Passo 2	S, n_2, n_3, n_4, n_1	S, n_2, n_3	(S,∞)	$(n_3,1)$	$(S,2)$	$(n_2,1)$	$(n_2,1)$	-
Passo 2	S, n_2, n_3, n_4, n_1, D	S, n_2, n_3, n_4	(S,∞)	$(n_3,1)$	$(S,2)$	$(n_2,1)$	$(n_2,1)$	$(n_4,1)$

punto si ripete la procedura con i passi indicati in Tabella 11.3. Il valore Δ è pari a 1 ed il cammino orientato in $G(\overline{X})$ da S a D è il seguente:

$$S \to n_2 \to n_4 \to D.$$

Ciò porta al seguente aggiornamento del flusso:

$$\overline{x}_{Sn_1} = 3 \;\; \overline{x}_{Sn_2} = 1 \;\; \overline{x}_{n_1n_3} = 1 \;\; \overline{x}_{n_1n_4} = 2$$

$$\overline{x}_{n_2n_3} = 0 \;\; \overline{x}_{n_2n_4} = 1 \;\; \overline{x}_{n_3D} = 1 \;\; \overline{x}_{n_4D} = 3$$

Il grafo $G(\overline{X})$ associato al nuovo flusso è illustrato in Figura 11.17. A questo punto si ripete la procedura con i passi indicati in Tabella 11.4. Il valore Δ è pari a 1 ed il cammino orientato in $G(\overline{X})$ da S a D è il seguente:

$$S \to n_2 \to n_3 \to n_1 \to n_4 \to D.$$

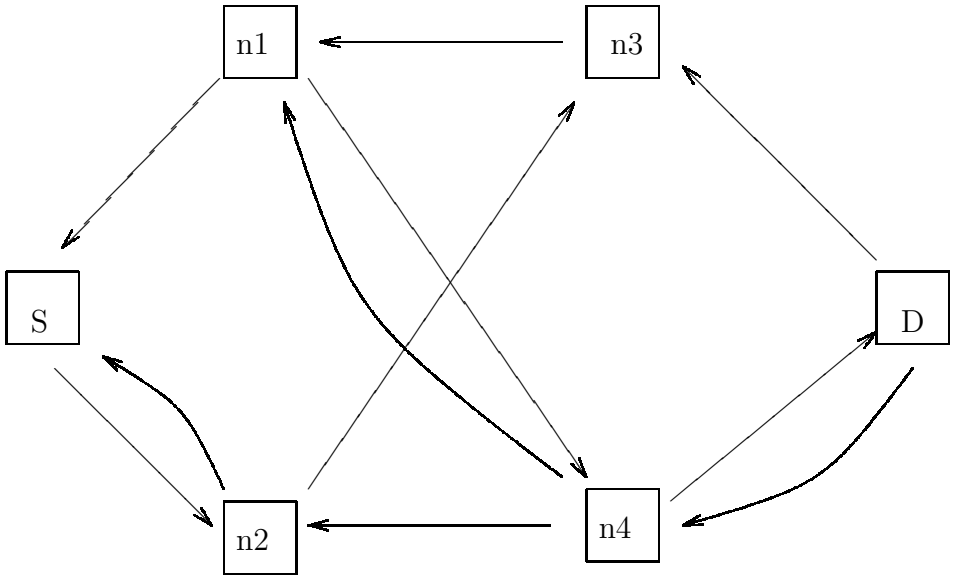


Figura 11.17:

Ciò porta al seguente aggiornamento del flusso:

$$\overline{x}_{Sn_1} = 3 \;\; \overline{x}_{Sn_2} = 2 \;\; \overline{x}_{n_1n_3} = 0 \;\; \overline{x}_{n_1n_4} = 3$$

$$\overline{x}_{n_2n_3} = 1 \;\; \overline{x}_{n_2n_4} = 1 \;\; \overline{x}_{n_3D} = 1 \;\; \overline{x}_{n_4D} = 4$$

Il grafo $G(\overline{X})$ associato al nuovo flusso è illustrato in Figura 11.18. A questo punto si ripete la procedura con i passi indicati in Tabella 11.5. Arriviamo a $E \setminus R = \emptyset$ e quindi possiamo fermarci ed affermare che il flusso attuale è quello ottimo. Non solo, sappiamo che il taglio indotto dal sottinsieme di nodi $E = \{S\}$ è quello a costo minimo. Si può infatti verificare che il valore del flusso uscente da S e il costo del taglio indotto da $\{S\}$ sono entrambi pari a 5.

Tabella 11.4:

Passo	E	R	S	n_1	n_2	n_3	n_4	D
1	S	$\emptyset$	(S,∞)	-	-	-	-	-
2	S, n_2	S	(S,∞)	-	$(S,1)$	-	-	-
2	S, n_2, n_3	S, n_2	(S,∞)	-	$(S,1)$	$(n_2,1)$	-	-
2	S, n_2, n_3, n_1	S, n_2, n_3	(S,∞)	$(n_3,1)$	$(S,1)$	$(n_2,1)$	-	-
2	S, n_2, n_3, n_1, n_4	S, n_2, n_3, n_1	(S,∞)	$(n_3,1)$	$(S,1)$	$(n_2,1)$	$(n_1,1)$	-
2	S, n_2, n_3, n_1, n_4, D	S, n_2, n_3, n_1, n_4	(S,∞)	$(n_3,1)$	$(S,1)$	$(n_2,1)$	$(n_1,1)$	$(n_4,1)$

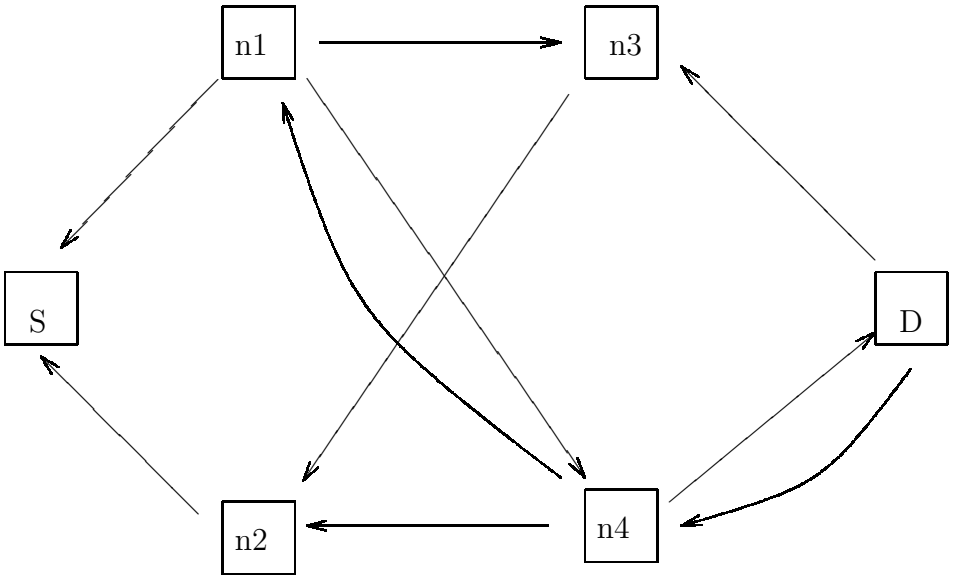


Figura 11.18:

Tabella 11.5:

	E	R	S	n_1	n_2	n_3	n_4	D
Passo 1	S	$\emptyset$	(S, ∞)	-	-	-	-	-
Passo 2	S	S	(S, ∞)	-	-	-	-	-

Capitolo 12

Un Branch-and-Bound per il TSP simmetrico

Nel Capitolo 9 abbiamo discusso i principi generali degli algoritmi branch-and-bound e abbiamo presentato un algoritmo branch-and-bound per problemi di PLI generici. In questo capitolo presenteremo un possibile approccio branch-and-bound per il problema TSP simmetrico ($d_{ij} = d_{ji}\ \forall\ i, j \in V$), mostrando in particolare come la struttura del problema suggerisca la possibilità di calcolare lower bound con un procedimento diverso dalla risoluzione di rilassamenti lineari e un metodo alternativo per l'esecuzione dell'operazione di branching.

12.1 Un lower bound per il caso simmetrico

Cominciamo presentando un rilassamento e, quindi, un modo di calcolare un lower bound per il problema del TSP simmetrico.

12.1.1 Un rilassamento del problema TSP simmetrico

Supponiamo di avere un grafo $G = (V, A)$ non orientato e con distanze d_{ij} sugli archi $(i, j) \in A$. Vogliamo calcolare un lower bound per il problema TSP simmetrico su tale grafo. Per arrivare a questo introduciamo dapprima la definizione di *1-tree*.

Definizione 10 *Dato un grafo $G = (V, A)$ non orientato e un suo nodo $a \in V$, chiamiamo 1-tree un sottografo $Q = (V, A_Q)$ di G con le seguenti proprietà:*

- *in A_Q ci sono esattamente due archi incidenti sul nodo a;*
- *se escludo da Q il nodo a e i due archi incidenti su di esso, mi rimane un albero sull'insieme di nodi $V \setminus \{a\}$.*

In particolare, da questa definizione segue che $\mid A_Q \mid = \mid V \mid$.

Esempio 40 *Dato il grafo con* $V = \{a, b, c, d, e\}$ *e*

$$A = \{(a, b); (a, c); (b, c); (b, e); (c, d); (d, a); (d, e)\},$$

il sottografo con

$$A_Q = \{(a, b); (d, a); (b, c); (b, e); (d, e)\}$$

è un 1-tree.

Si può notare che ogni circuito hamiltoniano è un 1-tree. Infatti, in un circuito hamiltoniano su ogni nodo incidono esattamente due archi ed inoltre togliendo un nodo a qualsiasi e i due archi del circuito incidenti su di esso si ottiene un albero sull'insieme di nodi $V \setminus \{a\}$. Il viceversa non è vero (lo 1-tree dell'esempio *non* è un circuito hamiltoniano). Qunidi se indichiamo con S' l'insieme degli 1-tree su un grafo G, tale insieme contiene la regione ammissibile S del problema `TSP`. In altre parole, il problema

$$\min_{Q=(V,A_Q)\in S'} \sum_{(i,j)\in A_Q} d_{ij}$$

risulta essere un rilassamento per il problema `TSP` simmetrico e la sua risoluzione restituisce un lower bound per il valore ottimo del problema `TSP`.

12.1.2 Calcolo del lower bound per il problema originario

Si pone ora il problema di come risolvere il rilassamento. Possiamo utilizzare la seguente procedura.

Passo 1. Si risolva il problema `MST` sul grafo ottenuto scartando da $G = (V, A)$ il nodo a prescelto e tutti gli archi incidenti su di esso. Sia A_T l'insieme di archi della soluzione trovata;

Passo 2. Si aggiungano ad A_T i due archi (a, k) e (a, h) a distanza minima tra tutti quelli incidenti sul nodo a prescelto.

Passo 3. Si restituisca $Q = (V, A_Q)$ con $A_Q = A_T \cup \{(a, k); (a, h)\}$.

Si noti come i tempi di calcolo siano polinomiali dal momento che la risoluzione del problema `MST` si può fare in tempo polinomiale con uno degli algoritmi visti nella Sezione 10.1.3 e lo stesso vale per il calcolo dei due valori minimi. Si noti anche che la scelta del nodo a è arbitraria. Al costo di un maggiore sforzo computazionale, si possono anche calcolare $\mid V \mid$ diversi lower bound scegliendo come nodo a tutti i nodi del grafo G e calcolando per ciascuno di essi il lower bound: come lower bound complessivo del problema originario si utilizza il migliore (ovvero il più grande) tra tutti i $\mid V \mid$ lower bound calcolati.

12.1.3 Calcolo del lower bound per sottoproblemi

Vogliamo qui definire la procedura di calcolo del lower bound per sottoproblemi di forma particolare. Dati $A_0, A_1 \subseteq A$, $A_0 \cap A_1 = \emptyset$, definiamo

$$S(A_0, A_1) = \{C = (V, A_C) \in S \ : \ x_{ij} = 0 \ \forall \ (i,j) \in A_0, \ x_{ij} = 1 \ \forall \ (i,j) \in A_1\}.$$

ovvero $S(A_0, A_1)$ contiene tutti i circuiti hamiltoniani che sicuramente non contengono gli archi in A_0 e sicuramente contengono gli archi in A_1. Per il calcolo del lower bound di un sottoproblema su $S(A_0, A_1)$, si utilizza la stessa procedura vista per il probelma `TSP` originario imponendo però la presenza degli archi in A_1 ed escludendo quella degli archi in A_0 sia nella risoluzione del problema `MST` sia nell'individuazione dei due archi incidenti sul nodo a. In particolare, si può risolvere il problema `MST` ad esempio con l'algoritmo greedy ma:

- inizializzando l'insieme di archi A_T non con l'insieme vuoto ma con tutti gli archi in A_1 non incidenti sul nodo a;
- non considerando gli archi in A_0 durante l'esecuzione dell'algoritmo greedy.

Inoltre:

- se in A_1 non sono presenti archi incidenti sul nodo a, metteremo in A_Q i due archi a distanza minima tra tutti quelli incidenti sul nodo a e al di fuori di A_0;
- se in A_1 è già presente un arco incidente sul nodo a questo entrerà in A_Q insieme a quello a distanza minima tra tutti quelli incidenti sul nodo a e al di fuori di A_0 e A_1;
- se in A_1 sono già presenti due archi incidenti sul nodo a, solo questi entreranno in A_Q.

Esempio 41 *Supponiamo di avere il seguente problema del* `TSP` *simmetrico*

	1	2	3	4	5
1	–	5	8	3	5
2	5	–	4	6	2
3	8	4	–	10	3
4	3	6	10	–	1
5	5	2	3	1	–

Proviamo a calcolare il lower bound per il sottoproblema $S(A_0, A_1)$ *con* $A_0 = \{(1,3);(4,5)\}$ *e* $A_1 = \{(1,5);(2,4)\}$. *Utilizziamo come nodo a il nodo 1. Per prima cosa dobbiamo risolvere il problema* `MST` *sull'insieme di nodi* $V \setminus \{1\}$ *imponendo la presenza dell'arco* $(2,4)$ *che è in* A_1 *ed escludendo quella degli archi in* A_0. *Utilizzando l'algoritmo greedy con* A_T *inizializzato con gli archi in* A_1 *non incidenti sul nodo 1 (in questo caso il solo arco* $(2,4)$*) ed escludendo la possibilità di inserire gli archi in* A_0*, arriviamo al seguente albero su* $V \setminus \{1\}$

$$A_T = \{(2,4);(2,5);(3,5)\}.$$

Notiamo che in A_1 è presente l'arco $(1,5)$ incidente sul nodo 1. Ad A_T dobbiamo quindi aggiungere, oltre a questo arco $(1,5)$, l'arco a distanza minima tra tutti quelli incidenti sul nodo 1 e al di fuori di A_0 e A_1, ovvero $(1,4)$. Quindi lo 1-tree ottimo ha l'insieme di archi

$$A_Q = \{(2,4); (2,5); (3,5); (1,5); (1,4)\}$$

con valore ottimo (e quindi lower bound per il sottoproblema $S(A_0, A_1)$) pari a 19. Si noti anche come $Q = (V, A_Q)$ non sia un circuito hamiltoniano e quindi non possa essere utilizzato per aggiornare (eventualmente) il valore di upper bound.

12.1.4 Un rilassamento lagrangiano

Vediamo ora di reinterpretare in altro modo il rilassamento basato sugli 1-tree e di indicare poi una strada per eventualmente migliorare i lower bound. Per prima cosa diamo il modello matematico del problema 1-tree. In esso avremo:

- una variabile binaria x_{ij} per ogni arco (i,j) (con valore 1 se l' arco viene inserito nello 1-tree e 0 altrimenti);
- un vincolo che impone che ci siano esattamente due archi incidenti sul nodo a:
$$\sum_{i \in V,\ i \neq a} x_{ia} = 2;$$
- vincoli che impongano che gli archi incidenti sui soli nodi in $V \setminus \{a\}$ formino un albero.

Per imporre che gli archi selezionati formino un albero su $V \setminus \{a\}$ dobbiamo richiedere che il numero di tali archi sia pari a $\mid V \setminus \{a\} \mid -1$, ovvero pari a $\mid V \mid -2$:

$$\sum_{i,j \in V \setminus \{a\}} x_{ij} = \mid V \mid -2;$$

e che tali archi non formino cicli. Per l'eliminazione di cicli in $V \setminus \{a\}$ utilizzeremo i seguenti vincoli. Dato $U \subseteq V \setminus \{a\}$, sia

$$E(U) = \{(i,j) \ : \ i,j \in U\}$$

Osservando che un ciclo sui nodi in U dovrebbe contenere $\mid U \mid$ archi in $E(U)$, per eliminare cicli imporremo

$$\sum_{(i,j) \in E(U)} x_{ij} \leq \mid U \mid -1 \qquad \forall\ U \subseteq V \setminus \{a\}$$

Si noti che per $\mid U \mid \leq 2$ i vincoli risultanti sono banali e possono essere omessi. Riassumendo, il modello matematico del problema 1 -tree è il seguente:

$$\begin{array}{ll} \min & \sum_{i,j\in V,\ i<j} d_{ij}x_{ij} \\ & \sum_{i\in V,\ i\neq a} x_{ia} = 2 \\ & \sum_{(i,j)\in E(U)} x_{ij} \leq \mid U \mid -1 \quad \forall\ U \subseteq V \setminus \{a\}\ : \mid U \mid \geq 3 \\ & \sum_{i,j\in V\setminus\{a\}} x_{ij} = \mid V \mid -2 \\ & x_{ij} \in \{0,1\} \qquad\qquad \forall\ i,j \in V,\ i<j \end{array}$$

Esempio 42 *Problema* `TSP` *simmetrico con la seguente tabella di distanze:*

	1	2	3	4
1	–	12	9	14
2	12	–	8	9
3	9	8	–	1
4	14	9	1	–

Si fissi come nodo a il nodo 1. Il modello matematico risultante è il seguente:

$$\begin{array}{rc} \min & 12x_{12} + 9x_{13} + 14x_{14} + 8x_{23} + 9x_{24} + x_{34} \\ & x_{12} + x_{13} + x_{14} = 2 \\ & x_{23} + x_{24} + x_{34} \leq 2 \\ & x_{23} + x_{24} + x_{34} = 2 \\ & x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34} \in \{0,1\} \end{array}$$

Si noti come in questo caso il vincolo $x_{23} + x_{24} + x_{34} \leq 2$ *sia implicato dall'altro vincolo* $x_{23} + x_{24} + x_{34} = 2$ *e quindi può essere omesso.*

Si può dimostrare che un modello valido per il problema `TSP` simmetrico è identico a quello visto per il problema 1-tree ma con l'aggiunta che su *tutti* i nodi in V incidano esattamente due archi, ovvero:

$$\begin{array}{ll} \min & \sum_{i,j\in V,\ i<j} d_{ij}x_{ij} \\ & \sum_{i\in V,\ i\neq j} x_{ij} = 2 \qquad \forall\ j \in V \\ & \sum_{(i,j)\in E(U)} x_{ij} \leq \mid U \mid -1 \quad \forall\ U \subseteq V \setminus \{a\}\ : \mid U \mid \geq 3 \\ & \sum_{i,j\in V\setminus\{a\}} x_{ij} = \mid V \mid -2 \\ & x_{ij} \in \{0,1\} \qquad\qquad \forall\ i,j \in V,\ i<j \end{array}$$

Esempio 43 *Il modello per il* TSP *simmetrico dell'esempio sarà:*

$$\begin{aligned}
\min \quad & 12x_{12} + 9x_{13} + 14x_{14} + 8x_{23} + 9x_{24} + x_{34} \\
& x_{12} + x_{13} + x_{14} = 2 \\
& x_{12} + x_{23} + x_{24} = 2 \\
& x_{13} + x_{23} + x_{34} = 2 \\
& x_{14} + x_{24} + x_{34} = 2 \\
& x_{23} + x_{24} + x_{34} = 2 \\
& x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34} \in \{0, 1\}
\end{aligned}$$

In pratica possiamo vedere il problema 1-tree come il rilassamento del problema TSP simmetrico ottenuto omettendo da questo tutti i vincoli che richiedono che vi siano esattamente due archi incidenti su ogni nodo, tranne quello relativo al nodo a. Come già visto, l'omissione di vincoli può essere vista come un caso particolare di rilassamento lagrangiano in cui tutti i moltiplicatori di Lagrange sono fissati a 0 e viene allora naturale chiedersi cosa succede se prendiamo moltiplicatori di Lagrange diversi da 0. Quindi, introduciamo ora moltiplicatori di Lagrange $\lambda = (\lambda_k)_{k \in V \setminus \{a\}}$ per i vincoli che richiedono che esattamente due archi incidano sui nodi, con l'unica eccezione del nodo a selezionato. Il modello risultante del rilassamento lagrangiano è il seguente:

$$\begin{aligned}
u(\lambda) \quad = \min \quad & \textstyle\sum_{i,j \in V,\ i<j} d_{ij} x_{ij} + \sum_{k \in V \setminus \{a\}} \lambda_k (2 - \sum_{i \in V,\ i \neq k} x_{ik}) & \\
& \textstyle\sum_{i \in V,\ i \neq a} x_{ia} = 2 & \\
& \textstyle\sum_{(i,j) \in E(U)} x_{ij} \leq \mid U \mid -1 & \forall\ U \subseteq V \setminus \{a\}\ : \mid U \mid \geq 3 \\
& \textstyle\sum_{i,j \in V \setminus \{a\}} x_{ij} = \mid V \mid -2 & \\
& x_{ij} \in \{0, 1\} & \forall\ i, j \in V,\ i < j
\end{aligned}$$

Si rammenti che essendo i vincoli di uguaglianza, possiamo considerare valori dei moltiplicatori di Lagrange $\lambda_k,\ k \in V \setminus \{a\}$, anche negativi e non solo maggiori o uguali a zero. Si vede che il rilassamento visto prima basato sul calcolo dello 1-tree minimo è un caso particolare di questo rilassamento lagrangiano in cui $\lambda_k = 0$ per tutti i $k \in V \setminus \{a\}$. Per comodità di notazione si include nell'obiettivo anche un termine relativo al vincolo di incidenza di esattamente due archi sul nodo a con il relativo moltiplicatore di Lagrange λ_a, imponendo però che questo possa assumere il solo valore 0. In tal modo avremo a che fare con un vettore di moltiplicatori di Lagrange $\lambda = (\lambda_k)_{k \in V}$ le cui componenti possono assumere valori positivi, negativi o nulli con la sola eccezione della componente relativa al nodo a che può assumere solo valore nullo.

Il modello del rilassamento lagrangiano può essere riscritto nel seguente modo:

$$
\begin{array}{lll}
u(\lambda) = \min & \sum_{i,j\in V,\ i<j} d_{ij}x_{ij} + \sum_{k\in V} \lambda_k(2 - \sum_{i\in V,\ i\neq k} x_{ik}) & \\
& \sum_{i\in V,\ i\neq a} x_{ia} = 2 & \\
& \sum_{(i,j)\in E(U)} x_{ij} \leq \mid U \mid -1 & \forall\ U \subseteq V \setminus \{a\}\ : \mid U \mid \geq 3 \\
& \sum_{i,j\in V\setminus\{a\}} x_{ij} = \mid V \mid -2 & \\
& x_{ij} \in \{0,1\} \qquad \forall\ i,j \in V,\ i<j &
\end{array}
$$

Un'ulteriore elaborazione del modello ci porta a:

$$
\begin{array}{lll}
u(\lambda) = \min & \sum_{i,j\in V,\ i<j}(d_{ij} - \lambda_i - \lambda_j)x_{ij} + 2\sum_{k\in V} \lambda_k & \\
& \sum_{i\in V,\ i\neq a} x_{ia} = 2 & \\
& \sum_{(i,j)\in E(U)} x_{ij} \leq \mid U \mid -1 & \forall\ U \subseteq V \setminus \{a\}\ : \mid U \mid \geq 3 \\
& \sum_{i,j\in V\setminus\{a\}} x_{ij} = \mid V \mid -2 & \\
& x_{ij} \in \{0,1\} \qquad \forall\ i,j \in V,\ i<j &
\end{array}
$$

Ma come possiamo risolvere il rilassamento lagrangiano? Per valori fissati dei moltiplicatori di Lagrange λ_k, $k \in V$, il rilassamento lagrangiano è facilmente risolvibile con la procedura vista in precedenza per l'individuazione dello 1-tree minimo. Infatti, il problema consiste nell'individuare lo 1-tree minimo tenendo conto che le distanze degli archi sono ora definite come segue

$$d'_{ij} = d_{ij} - \lambda_i - \lambda_j.$$

Esempio 44 *Il rilassamento lagrangiano può essere scritto in questa forma:*

$$
\begin{array}{ll}
u(\lambda_1,\ldots,\lambda_4) = \min & 12x_{12} + 9x_{13} + 14x_{14} + 8x_{23} + 9x_{24} + x_{34} + \\
& \lambda_1(2 - x_{12} - x_{13} - x_{14}) + \lambda_2(2 - x_{12} - x_{23} - x_{24}) + \\
& +\lambda_3(2 - x_{13} - x_{23} - x_{34}) + \lambda_4(2 - x_{14} - x_{24} - x_{34}) \\
& \\
& x_{12} + x_{13} + x_{14} = 2 \\
& x_{23} + x_{24} + x_{34} = 2 \\
& x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34} \in \{0,1\}
\end{array}
$$

Elaborando ulteriormente arriviamo a

$$
\begin{array}{ll}
u(\lambda_1,\ldots,\lambda_4) = \min & (12 - \lambda_1 - \lambda_2)x_{12} + (9 - \lambda_1 - \lambda_3)x_{13} + \\
& +(14 - \lambda_1 - \lambda_4)x_{14} + (8 - \lambda_2 - \lambda_3)x_{23} + \\
& +(9 - \lambda_2 - \lambda_4)x_{24} + (1 - \lambda_3 - \lambda_4)x_{34} + 2\sum_{i=1}^{4} \lambda_i \\
& \\
& x_{12} + x_{13} + x_{14} = 2 \\
& x_{23} + x_{24} + x_{34} = 2 \\
& x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34} \in \{0,1\}
\end{array}
$$

Una volta visto il rilassamento lagrangiano, possiamo introdurre anche il duale lagrangiano. Questo consisterà nell'individuare i valori $\lambda^* = (\lambda_k^*)_{k \in V}$, con $\lambda_a^* = 0$, per cui la funzione $u(\lambda)$ abbia il valore più grande possibile. In altre parole si tratta di risolvere il seguente problema

$$\max_{\lambda:\ \lambda_a = 0} u(\lambda)$$

Esempio 45 *Risolvendo il rilassamento lagrangiano con* $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$ *si ottiene lo 1-tree minimo*

$$(2,3)\ \ (3,4)\ \ (1,2)\ \ (1,3)$$

con un lower bound pari a 30. Se ora però consideriamo i seguenti moltiplicatori di Lagrange:

$$\lambda_1 = 0 \quad \lambda_2 = 0 \quad \lambda_3 = -1 \quad \lambda_4 = 1,$$

arriviamo ad un problema con la seguente tabella di distanze

	1	2	3	4
1	–	12	10	13
2	12	–	9	8
3	10	9	–	1
4	13	8	1	–

Lo 1-tree minimo con queste distanze è

$$(3,4)\ \ (2,4)\ \ (1,2)\ \ (1,3)$$

e il lower bound è pari a

$$2\sum_{i \in V} \lambda_i + (\textit{valore 1-tree minimo}) = 0 + (1 + 8 + 12 + 10) = 31,$$

migliore rispetto al precedente. Nel caso specifico si osserva anche che quest'ultimo 1-tree minimo è anche un circuito hamiltoniano con distanza complessiva pari a 31 ed è quindi soluzione ottima del problema in questione.

Non ci addentreremo nelle tecniche di risoluzione del duale lagrangiano ma diamo una possibile strategia per migliorare quanto ottenuto con determinati valori $\tilde{\lambda}_i$ dei moltiplicatori di Lagrange (ad esempio, $\tilde{\lambda}_i = 0$ per ogni i).

Nella soluzione ottenuta con i moltiplicatori di Lagrange $\tilde{\lambda}_i$ dobbiamo penalizzare i nodi con più di due archi incidenti su di essi (in modo da ridurre tale numero di archi incidenti) e favorire i nodi con meno di due archi incidenti su di essi (in modo da farne crescere il numero). Per fare questo dobbiamo ridurre il valore dei moltiplicatori di Lagrange relativi ai nodi con grado superiore a 2 nella soluzione e accrescere quello dei nodi con grado inferiore a 2. Per questo possiamo aggiornare i moltiplicatori di Lagrange nel modo seguente:

$$\bar{\lambda}_i = \tilde{\lambda}_i + 2 - \text{grado di } i \text{ nello 1-tree minimo.}$$

Esempio 46 *Nell'esempio considerato i gradi dei nodi 1, 2, 3 e 4 nello 1-tree minimo ottenuto con tutti i moltiplicatori di Lagrange nulli sono rispettivamente 2, 2, 3 e 1 e la regola appena vista porta proprio ai moltiplicatori di Lagrange proposti precedentemente.*

12.2 Aggiornamento dell'upper bound e branching

Per quanto rigaurda l'aggiornamento dell'upper bound, ricordiamo che questo si basa sull'identificazione di soluzioni ammissibili durante l'esecuzione dell'algoritmo. Nel nostro caso tale identificazione si può avere quando la soluzione ottima di un sottoproblema 1-tree dà origine a un circuito hamiltoniano: se la distanza di questo è inferiore all'upper bound attuale, possiamo aggiornare questo proprio con la distanza del circuito hamiltoniano individuato.

Ci occuperemo ora della regola di branching dell'algoritmo, ovvero di come un sottoproblema viene partizionato in sottoproblemi più piccoli. Cominceremo cercando di specificare come viene partizionata l'intera regione ammissibile iniziale S in piú sottinsiemi. Abbiamo visto che se non siamo nel caso fortunato in cui la soluzione del rilassamento é un circuito hamiltoniano, tale soluzione è uno 1-tree che contiene esattamente un sottocircuito. Forniremo una semplice regola di suddivisione il cui scopo é quello di impedire il formarsi nei nodi figli di tale sottocircuito. Indichiamo con $\{(i^1, j^1),\ (i^2, j^2), \ldots, (i^r, j^r)\}$ gli archi del sottocircuito. Il primo nodo figlio viene ottenuto imponendo che in esso non sia presente l'arco (i^1, j^1) (cioé si impone $x_{i^1,j^1} = 0$ e quindi avrà regione ammissibile di forma $S(A_0, A_1)$ con $A_0 = \{(i^1, j^1)\}$ e $A_1 = \emptyset$), il secondo nodo figlio viene ottenuto imponendo che sia presente l'arco (i^1, j^1) ma non sia presente l'arco (i^2, j^2) (cioé si impone $x_{i^1,j^1} = 1,\ x_{i^2,j^2} = 0$ e quindi avrà regione ammissibile di forma $S(A_0, A_1)$ con $A_0 = \{(i^2, j^2)\}$ e $A_1 = \{(i^1, j^1)\}$), e cosí via fino al r-esimo figlio in cui si impone che siano presenti gli archi (i^k, j^k), $k = 1, \ldots, r-1$, ma non sia presente l'arco (i^r, j^r) (cioé si impone $x_{i^k,j^k} = 1,\ k = 1, \ldots, r-1,\ x_{i^r,j^r} = 0$ e e quindi avrà regione ammissibile di forma $S(A_0, A_1)$ con $A_0 = \{(i^r, j^r)\}$ e $A_1 = \{(i^1, j^1), \ldots, (i^{r-1}, j^{r-1})\}$). Si veda la Figura 12.1. In altre parole si fa in modo che in ciascuno dei figli sia assente almeno un arco del sottocircuito il che esclude la presenza di tale sottocircuito in ciascuno dei nodi figli.

Notiamo che i nodi figli/sottoproblemi che vengono creati hanno tutti regione ammissibile di forma $S(A_0, A_1)$ con diversi sottinsiemi A_0 e A_1. Vediamo ora di illustrare la regola di branching per sottoproblemi con regione ammissibile $S(A_0, A_1)$. Indichiamo con:

$$A_s = \{(i^k, j^k),\ k = 1, \ldots, \ell\}.$$

l'insieme di archi che stanno nella soluzione del problema 1-tree ma non in A_1. Se l'insieme di archi $A_s \cup A_1$ forma un circuito hamiltoniano, cioè appartiene

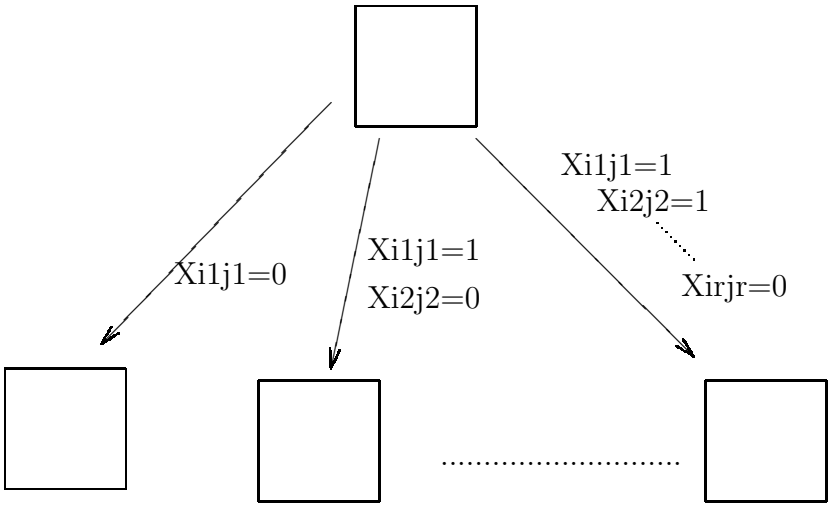


Figura 12.1: Branching per il problema TSP simmetrico.

alla regione ammissibile $S(A_0, A_1)$ e quindi anche a S, il suo valore coincide con il valore del lower bound trovato per il sottinsieme e possiamo utilizzare tale valore per aggiornare, eventualmente, l'upper bound UB (come già osservato in precedenza). Altrimenti l'insieme di archi $A_s \cup A_1$ conterrà un sottocircuito. A questo punto per effetuare il branching del sottinsieme $S(A_0, A_1)$ si ripete esattamente quanto già visto per il nodo radice. Quindi, se indichiamo con $\{(i^1, j^1),\ (i^2, j^2), \ldots, (i^t, j^t)\}$ gli archi del sottocircuito scelto che appartengono ad A_s, il primo nodo figlio viene ottenuto imponendo che in esso non sia presente l'arco (i^1, j^1) (cioé si impone $x_{i^1,j^1} = 0$), il secondo nodo figlio vine ottenuto imponendo che sia presente l'arco (i^1, j^1) ma non sia presente l'arco (i^2, j^2) (cioé si impone $x_{i^1,j^1} = 1,\ x_{i^2,j^2} = 0$), e cosí via fino al t-esimo figlio in cui si impone che siano presenti gli archi (i^k, j^k), $k = 1, \ldots, t-1$, ma non sia presente l'arco (i^t, j^t) (cioé si impone $x_{i^k,j^k} = 1,\ k = 1, \ldots, t-1,\ x_{i^t,j^t} = 0$). Va notato che i nuovi nodi figli creati hanno tutti regione ammissibile della forma $S(A_0, A_1)$ e questo giustifica il fatto che abbiamo indicato una regola di calcolo del lower bound solo per insiemi di questa forma.

Esempio 47 *Consideriamo l'Esempio 41. Avevamo in tale esempio il sotto-problema $S(A_0, A_1)$ con $A_0 = \{(1,3); (4,5)\}$ e $A_1 = \{(1,5); (2,4)\}$. Nel calcolo del lower bound avevamo individuato lo 1-tree $Q = (V, A_Q)$ con*

$$A_Q = \{(2,4); (2,5); (3,5); (1,5); (1,4)\}$$

In questo è presente il circuito

$$1 \to 4 \to 2 \to 5 \to 1.$$

Gli archi in A_s sono i seguenti

$$A_s = \{(1,4),\ (2,5)\}.$$

Ne consegue che l'operazione di branching genererà due nodi, un nodo $S_1(A_0, A_1)$ *con*

$$A_0 = \{(1,3);(4,5);(1,4)\} \qquad A_1 = \{(1,5);(2,4)\}$$

e l'altro nodo $S_2(A_0, A_1)$ *con*

$$A_0 = \{(1,3);(4,5);(2,5)\} \qquad A_1 = \{(1,5);(2,4);(1,4)\}$$

Capitolo 13

Programmazione dinamica

Per la risoluzione di problemi difficili abbiamo sino a ora discusso solo algoritmi di tipo branch-and-bound (con esempi per problemi di PLI generici e per problemi `TSP`) e di taglio (con esempi solo nell'ambito dei problemi di PLI generici). Quando i problemi in questione soddisfano determinate proprietà, è possibile applicare un altro tipo di approccio, la *programmazione dinamica*. Nel seguito indicheremo quali sono queste proprietà e come queste consentano di definire l'approccio di programmazione dinamica. Il tutto verrà illustrato tramite la derivazione di un approccio di programmazione dinamica per il problema `KNAPSACK`

13.1 Proprietà richieste per l'applicazione della programmazione dinamica

La programmazione dinamica è applicabile a problemi con le seguenti proprietà.

1. Il problema può essere suddiviso in n *blocchi*.

2. In ogni blocco k, $k = 1, \ldots, n$ ci si trova in uno degli *stati* s_k appartenenti all'insieme di stati $Stati_k$. L'insieme $Stati_1$ del blocco 1 è costituito da un singolo stato s_1.

3. In ogni blocco k si deve prendere una decisione d_k appartenente ad un insieme di possibili decisioni D_k. L'insieme di possibili decisioni può dipendere dallo stato s_k, ovvero $D_k = D_k(s_k)$.

4. Se al blocco k si prende la decisione d_k e ci si trova nello stato s_k, il blocco k fornisce un *contributo* alla funzione obiettivo f del problema pari a $u(d_k, s_k)$. La funzione obiettivo f sarà pari alla somma dei contributi degli n blocchi (esistono anche varianti in cui i contributi non si sommano ma si moltiplicano tra loro ma qui ne omettiamo la trattazione).

5. Se al blocco k ci si trova nello stato s_k e si prende la decisione d_k, al blocco $k+1$ ci troveremo nello stato $s_{k+1} = t(d_k, s_k)$. La funzione t viene detta *funzione di transizione* (la Figura 13.1 illustra la transizione).

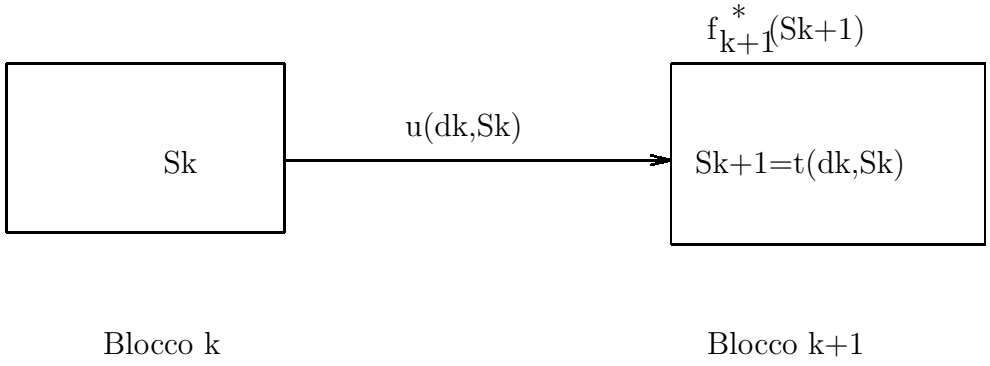


Figura 13.1: La transizione dallo stato s_k del blocco k allo stato s_{k+1} del blocco $k+1$ quando si prende la decisione d_k.

6. Infine, la proprietà essenziale è quella che viene chiamata *principio di ottimalità*: se al blocco k mi trovo nello stato s_k, la sequenza di decisioni ottime da prendere nei blocchi $k, k+1, \ldots, n$ è *totalmente indipendente* da come sono giunto allo stato s_k, ovvero dalle decisioni ai blocchi $1, \ldots, k-1$ che mi hanno fatto arrivare allo stato s_k.

Poichè il principio di ottimalità sancisce che la sequenza di decisioni ottime a partire da una stato s_k al blocco k fino al blocco n è indipendente da come sono arrivato allo stato s_k, posso definire una funzione $f_k^*(s_k)$ che restituisce il valore ottimo delle somme dei contributi dei blocchi $k, k+1, \ldots, n$ quando si parte dallo stao s_k al blocco k. Tipicamente è semplice calcolare $f_n^*(s_n)$ per ogni $s_n \in Stati_n$, cioè il valore ottimo del solo contributo del blocco n quando ci si trova nello stato s_n. La corrispondente decisione ottima verrà indicata con $d_n^*(s_n)$. A questo punto posso procedere a ritroso per calcolare $f_{n-1}^*(s_{n-1})$ per ogni $s_{n-1} \in Stati_{n-1}$. Infatti, dato un generico stato s_{n-1} in $Stati_{n-1}$, consideriamo una generica decisione $d_{n-1} \in D_{n-1}$. Il contributo di tale decisione al blocco $n-1$ è pari a $u(d_{n-1}, s_{n-1})$. Inoltre mi sposto nello stato $s_n = t(d_{n-1}, s_{n-1})$ al blocco n. Da qui posso procedere in modo ottimo e quindi con un contributo pari a $f_n^*(t(d_{n-1}, s_{n-1}))$. Quindi, se mi trovo nello stato s_{n-1} e prendo la decisione d_{n-1} il contributo complessivo dei blocchi $n-1$ e n sarà dato da

$$u(d_{n-1}, s_{n-1}) + f_n^*(t(d_{n-1}, s_{n-1})).$$

Tra tutte le possibili decisioni d_{n-1} la migliore sarà quella che rende massimo il contributo complessivo. Tale decisione viene indicata con $d_{n-1}^*(s_{n-1})$ e si avrà

$$f_{n-1}^*(s_{n-1}) = u(d_{n-1}^*(s_{n-1}), s_{n-1}) + f_n^*(t(d_{n-1}^*(s_{n-1}), s_{n-1})) =$$

$$= \max_{d_{n-1} \in D_{n-1}} [u(d_{n-1}, s_{n-1}) + f_n^*(t(d_{n-1}, s_{n-1}))].$$

Una volta calcolati i valori $f_{n-1}^*(s_{n-1})$ per tutti gli stati $s_{n-1} \in Stati_{n-1}$, possiamo continuare a procedre a ritroso. Per il blocco k avremo

$$\forall\ s_k \in Stati_k : \quad f_k^*(s_k) = \max_{d_k \in D_k} [u(d_k, s_k) + f_{k+1}^*(t(d_k, s_k))],$$

con la corrispondente decisione ottima $d_k^*(s_k)$ (si noti che poichè si procede a ritroso i valori f_{k+1}^* sono già stati calcolati). Arrivati al blocco 1 si ha che $Stati_1$ è formato da un unico stato s_1 e il valore $f_1^*(s_1)$ coincide con il valore ottimo del problema. Per ricostruire la soluzione ottima possiamo partire dal blocco 1. Al blocco 1 la decisione ottima è $d_1^*(s_1)$. Con tale decisione ci spostiamo allo stato $s_2^* = t(d_1^*(s_1), s_1)$ del blocco 2 e la decisione ottima per tale blocco sarà $d_2^*(s_2^*)$. Con tale decisione ci spostiamo allo stato $s_3^* = t(d_2^*(s_2^*), s_2^*)$ del blocco 3 e la decisione ottima per tale blocco sarà $d_3^*(s_3^*)$. Si procede in questo modo fino ad arrivare al blocco n. Riassumendo, avremo il seguente algoritmo.

Passo 1 Per ogni $s_n \in Stati_n$ si calcoli $f_n^*(s_n)$ e la corrispondente decisione ottima $d_n^*(s_n)$. Si ponga $k = n - 1$.

Passo 2 Per ogni $s_k \in Stati_k$ si calcoli

$$f_k^*(s_k) = \max_{d_k \in D_k} [u(d_k, s_k) + f_{k+1}^*(t(d_k, s_k))]$$

e la corrispondente decisione ottima $d_k^*(s_k)$.

Passo 3 Se $k = 1$ si ha che $f_1^*(s_1)$ è il valore ottimo del problema. Altrimenti si ponga $k = k - 1$ e si ritorni al Passo 2.

La soluzione ottima può essere ricostruita attraverso la seguente procedura.

Passo 1 Si ponga $s_1^* = s_1$ e $k = 1$.

Passo 2 Per il blocco k la decisione ottima è $d_k^*(s_k^*)$. Si ponga

$$s_{k+1}^* = t(d_k^*(s_k^*), s_k^*).$$

Passo 3 Se $k = n$, stop: la soluzione ottima è stata ricostruita ed è rappresentata dalle decisioni

$$d_1^*(s_1^*), \ldots, d_n^*(s_n^*).$$

Altrimenti si ponga $k = k + 1$ e si ritorni al Passo 2.

Vedremo ora un esempio di algoritmo di programmazione dinamica applicato al problema dello zaino.

13.2 Un algoritmo di programmazione dinamica per il problema dello zaino

Prima di tutto dobbiamo definire i blocchi, gli stati, le decisioni possibili, i contributi di ogni blocco e la funzione di transizione.

1. I blocchi coincidono con gli oggetti. Avremo quindi un blocco per ogni oggetto.

2. Al blocco k i possibili stati s_k sono i valori $0, 1, \ldots, b$ che rappresentano la capacità residua dello zaino per i blocchi da k fino a n. Quindi avremo

$$Stati_k = \{0, 1, \ldots, b\}$$

   per $k = 2, \ldots, n$, mentre $Stati_1$ contiene il solo stato b (la capacità iniziale dello zaino).

3. Le decisioni al blocco k possono essere due se $p_k \leq s_k$ (ovvero il peso dell'oggetto k non supera la capacità residua dello zaino) oppure una sola se $p_k > s_k$. Nel primo caso le due decisioni sono NO (non inserire l'oggetto k nello zaino) oppure SI (inserire l'oggetto k nello zaino), nel secondo caso la sola decisione possibile è NO.

4. Il contributo al blocco k sarà dato dal valore dell'oggetto se la decisione è SI, mentre sarà nullo se la decisione è NO. Quindi, per ogni s_k si avrà

$$u(NO, s_k) = 0 \quad u(SI, s_k) = v_k.$$

5. La funzione di transizione pone $s_{k+1} = s_k$ se la decisione è NO, e $s_{k+1} = s_k - p_k$ se la decisione è SI. Quindi

$$t(NO, s_k) = s_k \quad t(SI, s_k) = s_k - p_k.$$

6. Si noti che vale il principio di ottimalità. Infatti, se ad un blocco k ho una capacità residua dello zaino pari a s_k, le decisioni ottime circa gli oggeti $k, k+1, \ldots, n$ da inserire sono del tutto indipendenti da come sono giunto, attraverso le decisioni ai blocchi $1, \ldots, k-1$, ad avere la capacità residua s_k.

7. I valori $f_n^*(0), \ldots, f_n^*(b)$ sono facilmente calcolabili. Infatti si ha

$$f_n^*(s_n) = v_n, \quad d_n^*(s_n) = SI \quad \forall\ s_n \geq p_n$$

$$f_n^*(s_n) = 0, \quad d_n^*(s_n) = NO \quad \forall\ s_n < p_n.$$

A questo punto siamo pronti ad applicare l'algoritmo all'esempio con i seguenti dati: $b = 16$ e

i	1	2	3	4
v_i	8	6	10	1
p_i	7	7	13	4

Tabella 13.1: La funzione f_4^* con le relative decisioni ottime

s_4	f_4^*	d_4^*
0	0	NO
1	0	NO
2	0	NO
3	0	NO
4	1	SI
5	1	SI
6	1	SI
7	1	SI
8	1	SI
9	1	SI
10	1	SI
11	1	SI
12	1	SI
13	1	SI
14	1	SI
15	1	SI
16	1	SI

Tabella 13.2: La funzione f_3^* con le relative decisioni ottime

s_3	f_3^*	d_3^*
0	0	NO
1	0	NO
2	0	NO
3	0	NO
4	1	NO
5	1	NO
6	1	NO
7	1	NO
8	1	NO
9	1	NO
10	1	NO
11	1	NO
12	1	NO
13	$\max\{1, 10+0\} = 10$	SI
14	$\max\{1, 10+0\} = 10$	SI
15	$\max\{1, 10+0\} = 10$	SI
16	$\max\{1, 10+0\} = 10$	SI

Tabella 13.3: La funzione f_2^* con le relative decisioni ottime

s_2	f_2^*	d_2^*
0	0	NO
1	0	NO
2	0	NO
3	0	NO
4	1	NO
5	1	NO
6	1	NO
7	$\max\{1, 6+0\} = 6$	SI
8	$\max\{1, 6+0\} = 6$	SI
9	$\max\{1, 6+0\} = 6$	SI
10	$\max\{1, 6+0\} = 6$	SI
11	$\max\{1, 6+1\} = 7$	SI
12	$\max\{1, 6+1\} = 7$	SI
13	$\max\{10, 6+1\} = 10$	NO
14	$\max\{10, 6+1\} = 10$	NO
15	$\max\{10, 6+1\} = 10$	NO
16	$\max\{10, 6+1\} = 10$	NO

Costruiamo dapprima la Tabella 13.1 per il blocco 4. Con questa tabella possiamo costruire la Tabella 13.2 per il blocco 3. Con la Tabella 13.2 possiamo costruire la Tabella 13.3 per il blocco 2. Resta ora solo da calcolare $f_1^*(16)$ e la relativa decisone ottima. Si ha:

$$f_1^*(16) = \max\{10, 8+6\} = 14 \quad d_1^*(16) = SI.$$

Possiamo quindi concludere che il valore ottimo del nostro problema è 14. Vediamo ora di ricostruire anche la soluzione ottima. Si ha:

$$\begin{aligned} d_1^*(16) &= SI \quad s_2^* = t(SI, 16) = 9 \\ d_2^*(s_2^*) &= SI \quad s_3^* = t(SI, 9) = 2 \\ d_3^*(s_3^*) &= NO \quad s_4^* = t(NO, 2) = 2 \\ d_4^*(s_4^*) &= NO \end{aligned}$$

Da cui si conclude che la soluzione ottima si ottiene inserendo nello zaino gli oggetti 1 e 2, ovvero $N^* = \{1, 2\}$.

Si può dimostrare che la procedura di programmazione dinamica appena vista richiede un numero di operazioni pari a $O(nb)$. Tale numero di operazioni non rapppresenta una complessità polinomiale (perché?).

Capitolo 14

Algoritmi di approssimazione

Nella Sezione 10.2.5 abbiamo introdotto le definizioni di problemi e algoritmi di approssimazione, distinguendo i problemi in 4 classi sulla base della difficoltà dei corrispondenti problemi di ε-approssimazione. In questo capitolo discuteremo due problemi, uno appartenente alla terza classe (quella dei problemi in cui il problema di ε-approssimazione è risolvibile in tempo polinomiale per ε sufficientemente elevato, ma è $\mathcal{NP}$-completo per valori piccoli di ε) e uno appartenente alla prima classe, quella dei problemi che ammettono un FPTAS (Fully Polynomial Time Approximation Scheme, o schema di approssimazione completamente polinomiale).

14.1 Un algoritmo di approssimazione per il problema TSP metrico

Il problema TSP *metrico* è un problema TSP simmetrico in cui si introducono delle restrizioni sui possibili valori che possono assumere le distanze lungo gli archi di un grafo. Come vedremo le restrizioni modificano (in particolare, riducono) la difficoltà del problema. Il problema TSP metrico comprende i problemi TSP simmetrici in cui le distanze d_{ij} degli archi sono tutte non negative ed inoltre soddisfano la diseguaglianza triangolare:

$$\forall\ i, j, k\ :\quad d_{ij} + d_{jk} \geq d_{ik} \tag{14.1}$$

Consideriamo ora il seguente algoritmo, denominato algoritmo *Double Spanning Tree*, abbreviato con DST nel seguito.

Algoritmo DST

Passo 1 Dato il grafo $G = (V, A)$, si determini un albero di supporto a costo minimo di tale grafo e lo si indichi con $T = (V, A_T)$.

Passo 2 Si duplichi ogni arco in A_T assegnando ad ogni arco duplicato la stessa distanza dell'arco originale. Sul grafo risultante si determini un *ciclo euleriano*, ovvero un ciclo che partendo da un nodo attraversi tutti gli archi del grafo una ed una sola volta.

Passo 3 Dalla sequenza di nodi

$$i_1 \to i_2 \to \cdots \to i_{2n-2} \to i_{2n-1} = i_1$$

si eliminino tutte le ripetizioni di nodi a parte quella dell'ultimo nodo. La sequenza di nodi risultante è un circuito hamiltoniano.

Si può dimostrare che l'algoritmo DST richiede tempo di esecuzione polinomiale rispetto alla dimensione dell'istanza. Si noti che un grafo ammette un ciclo euleriano se e solo se tutti i suoi nodi hanno grado pari. Nel nostro caso, avendo raddoppiato tutti gli archi dell'albero di supporto ottimo (e quindi i gradi dei nodi di tale albero), la condizione 'e ovviamente soddisfatta. Restya ancora da vedere come individuare un ciclo euleriano. Si può utilizzare una procedura che, fissato un nodo radice dell'albero, consiste in pratica in una visita in profondità dell'albero:

Algoritmo per determinare un ciclo euleriano

Passo 1 Dato l'albero $T = (V, A_T)$, si fissi un suo nodo radice v^* in modo arbitrario. Si ponga:

$$S = \emptyset, \quad i_1 = v^*, \quad k = 2, \quad w = v^*.$$

Passo 2 Se w ha nodi figli in $V \setminus S$, allora si selezioni un suo nodo figlio $z \in V \setminus S$, si ponga

$$w = z, \quad i_k = z, \quad k = k + 1$$

e si ripeta il Passo 2.
Altrimenti (cioè se w non ha nodi figli in $V \setminus S$): se $w \neq v^*$, si risalga al nodo padre y di w, si ponga

$$S = S \cup \{w\}, \quad w = y, \quad i_k = y, \quad k = k + 1$$

e si ripeta il Passo 2; altrimenti (se $w = v^*$) ci si arresti.

La procedura DST viene ora illustrata su un esempio.

Esempio 48 *Sia dato il grafo in Figura 14.1 con le distanze indicate su ciascun arco (si noti che ci troviamo nel caso simmetrico e quindi gli archi non sono orientati, il che significa che per ogni arco (i, j) la distanza per andare da i a j è uguale a quella per andare da j a i). Si può verificare che gli archi del grafo soddisfano la diseguaglianza triangolare e quindi il problema è un'istanza di problema di* `TSP` *metrico. Nel Passo 1 si deve determinare un albero di supporto a costo minimo per tale grafo. Tale albero (ottenuto ad esempio con la nota*

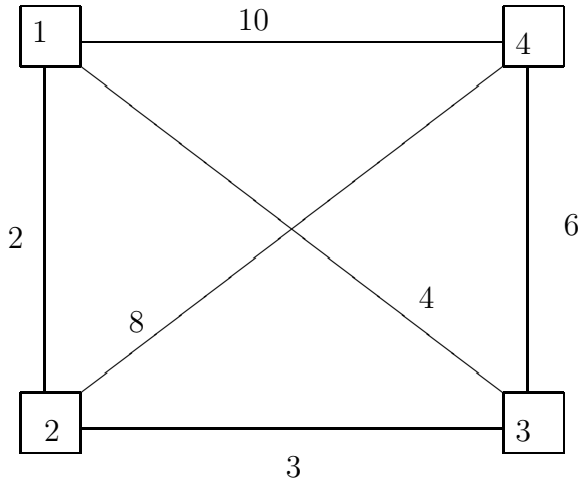


Figura 14.1:

procedura greedy) è indicato in Figura 14.2. Al Passo 2 si richiede di duplicare gli archi dell'albero (si veda la Figura 14.3) e di determinare un ciclo euleriano sul grafo ottenuto in questo modo. Partendo dal nodo 1 possiamo ottenere il seguente ciclo euleriano:

$$1 \to 2 \to 3 \to 4 \to 3 \to 2 \to 1.$$

Nel Passo 3 dobbiamo eliminare le ripetizioni di nodi (tranne quella relativa all'ultimo nodo) ed otteniamo il seguente circuito hamiltoniano:

$$1 \to 2 \to 3 \to 4 \to 1.$$

Il valore di tale circuito è 21. Si noti che non è quello ottimo: il circuito

$$1 \to 2 \to 4 \to 3 \to 1$$

ha valore pari a 20.

Come visto, l'algoritmo non è esatto per il problema del `TSP` metrico ma la seguente osservazione evidenzia che si tratta di un algoritmo di approssimazione per questo problema.

Osservazione 44 *Per il* `TSP` *metrico l'algoritmo DST è un algoritmo di* 1*-approssimazione.*

Dimostrazione
Sia $T = (V, A_T)$ l'albero di supporto a costo minimo individuato al Passo 1 dell'algoritmo e sia

$$i_1 \to i_2 \to \cdots \to i_{2n-2} \to i_{2n-1} = i_1 \tag{14.2}$$

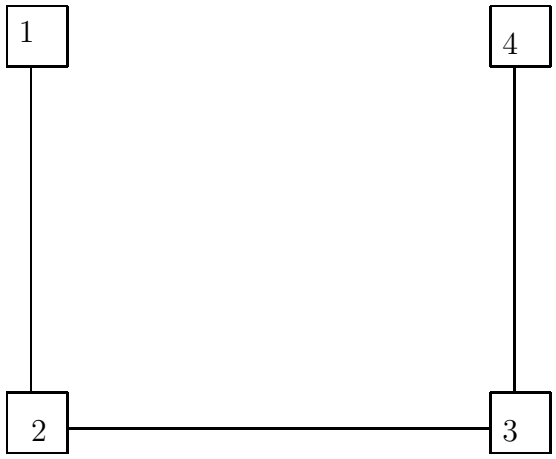


Figura 14.2:

il ciclo euleriano individuato al Passo 2 dell'algoritmo dopo aver raddoppiato gli archi dell'albero T. Indichiamo con

$$A_Q = \{(i_j, i_{j+1}) \ : \ j = 1, \ldots, 2n-2\}$$

l'insieme degli archi di tale ciclo. Si noti che

$$\sum_{e \in A_Q} d_e = 2 \sum_{e \in A_T} d_e. \tag{14.3}$$

Quando nel Passo 3. rimuoviamo un nodo i_j in quanto già presente, sostituiamo nel cammino la coppia di archi

$$(i_{j-1}, i_j) \quad (i_j, i_{j+1})$$

con il singolo arco

$$(i_{j-1}, i_{j+1}).$$

Ma per la diseguaglianza triangolare (14.1) si ha

$$d_{i_{j-1}, i_j} + d_{i_j, i_{j+1}} \geq d_{i_{j-1}, i_{j+1}}.$$

Ripetendo questo ragionamento per ogni nodo rimosso, si ha che il circuito hamiltoniano $C = (V, A_C)$

$$i'_1 \to \ldots \to i'_n \to i'_1$$

ottenuto dal ciclo euleriano (14.2) con l'eliminazione dei nodi ripetuti, ha distanza complessiva certamente non superiore a quella del ciclo euleriano. In base a (14.3) avremo dunque

$$\sum_{e \in A_C} d_e \leq \sum_{e \in A_Q} d_e = 2 \sum_{e \in A_T} d_e. \tag{14.4}$$

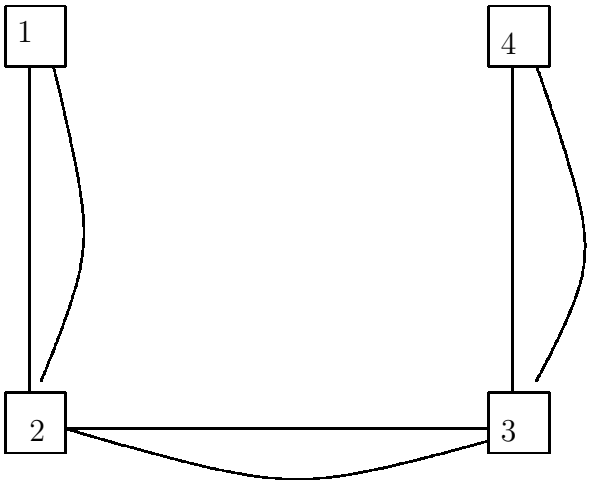


Figura 14.3:

Ma per la non negatività degli archi si ha che il circuito hamiltoniano $C^* = (V, A_{C^*})$ sluzione ottima del problema di TSP metrico, ha valore complessivo non inferiore a quello dell'albero di supporto a costo minimo, cioè

$$\sum_{e \in A_{C^*}} d_e \geq \sum_{e \in A_T} d_e. \tag{14.5}$$

Infatti rimuovendo un qualsiasi arco del circuito si riduce il valore complessivo dell'insieme di archi (per la non negatività del peso dell'arco rimosso) e si ottiene un grafo ancora connesso ed aciclico, ovvero un albero di supporto il cui valore non può essere inferiore a quello dell'albero T che ha costo minimo. Da (14.4) e (14.5) si ricava che

$$\sum_{e \in A_C} d_e \leq 2 \sum_{e \in A_T} d_e \leq 2 \sum_{e \in A_{C^*}} d_e,$$

o, equivalentemente,

$$\frac{\sum_{e \in A_C} d_e}{\sum_{e \in A_{C^*}} d_e} \leq 2,$$

e ciò dimostra che l'algoritmo DST è un algoritmo di 1-approssimazione per il problema TSP metrico.

Ci si può chiedere se è possibile fare di meglio per il problema TSP metrico, ovvero se esiste un algoritmo di ε-approssimazione con valore di $\varepsilon < 1$. Un tale algoritmo esiste, anche se non lo vedremo, ed il corrispondente valore di ε è 0.5. Si può anche però dimostrare il seguente risultato negativo.

Osservazione 45 *Per il problema* TSP *metrico esiste un* $\overline{\varepsilon} > 0$ *tale che per ogni* $\varepsilon \leq \overline{\varepsilon}$ *il problema di* ε*-approssimazione associato al* TSP *metrico è* $\mathcal{NP}$*-completo.*

Questa osservazione ci consente di collocare il problema TSP metrico nel Caso 3 tra quelli visti nella Sezione 10.2.5.

14.2 Un FPTAS per il problema KNAPSACK

Vogliamo ora definire un FPTAS (Fully Polynomial Time Approximation Scheme, ovvero uno schema di approssimazione completamente polinomiale) per il problema KNAPSACK. Questo è una classe di algoritmi di ε-approssimazione che risultano essere polinomiali sia rispetto alla dimensione del problema KNAPSACK sia rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. Prima di arrivare a definire un FPTAS abbiamo bisogno di introdurre un nuovo metodo esatto (ovviamente non polinomiale) di risoluzione per il problema.

14.2.1 Un metodo di risoluzione esatto alternativo

In alternativa al metodo di programmazione dinamica visto in precedenza per risolvere il problema KNAPSACK, si può utilizzare anche questo altro metodo, sempre di programmazione dinamica.

Inizializzazione Si ponga $M_0 = \{(\emptyset, 0)\}$. Si ponga $j = 1$.

Passo 1 Si ponga $M_j = \emptyset$.

Passo 2 Per ogni terna $(N, p, v) \in M_{j-1}$, aggiungi in M_j l'elemento (N, p, v) e, se $p_j + p \leq b$, aggiungi in M_j anche $(N \cup \{j\}, p + p_j, v + v_j)$.

Passo 3 Per ogni coppia di elementi (S, p, v) e (S', p', v') in M_j, se

$$p' \geq p \quad \text{e} \quad v' = v,$$

allora scarta la terna (S', p', v').

Passo 4 Se $j = n$, allora restituisci come soluzione ottima del problema KNAPSACK la coppia in M_n con il massimo valore della seconda componente, altrimenti poni $j = j + 1$ e torna al Passo 1.

Si noti che ad ogni iterazione ogni coppia dell'insieme M_j ha come prima componente una soluzione ammissibile del problema KNAPSACK contenente *solo* i primi j oggetti, come seconda componente il relativo peso e come terza componente il relativo valore dell'obiettivo. Inoltre, il Passo 3 esclude quelle soluzioni ammissibili che sono dominate da altre, ovvero con lo stesso valore dell'obiettivo (uguale terza componente) ma con peso complessivo (seconda componente) superiore o uguale a quello di altre soluzioni. Si noti quindi che in M_n ci sono tutte le soluzioni ammissibili contenenti tutti gli n oggetti e quella con seconda componente massima è quindi anche soluzione ottima del problema KNAPSACK.

Volendo essere più precisi, la regola

$$p' \geq p \quad \text{e} \quad v' = v,$$

utilizzata per scartare la terna (S', p', v'), può essere rafforzata come segue

$$p' \geq p \quad \text{e} \quad v' \leq v.$$

Tuttavia, nel seguito, per semplificare l'analisi dell'algoritmo, utilizzeremo la regola più debole in cui si richiede l'uguaglianza tra i valori.

Calcoliamo ora il numero di operazioni eseguite dalla procedura. Sia v^* il valore ottimo del problema.

Osservazione 1 *La procedura di risoluzione sopra descritta richiede un numero di operazioni $O(nv^*)$.*

Dimostrazione In ogni insieme M_{j-1} abbiamo al più v^* elementi (ve ne è al più uno per ogni possibile valore della seconda componente e i possibili valori della seconda componente sono al più v^*). L'operazione di aggiornamento al Passo 2 deve essere fatta su al più v^* coppie per ognuna delle quali si devono fare al più due somme. Quindi, in tutto sono richieste al più $O(v^*)$ operazioni. Al Passo 3 si devono individuare eventuali coppie di soluzioni con lo stesso valore dell'obiettivo e scartare quella con peso maggiore (o una delle due se hanno lo stesso peso). Essendoci in M_j al più $2v^*$ soluzioni, tale operazione si può implementare in modo da dover eseguire $O(v^*)$ operazioni. Tenuto conto che i passi dell'algoritmo devono essere ripetuti n volte, abbiamo un totale di $O(nv^*)$ operazioni.

14.2.2 Descrizione del FPTAS

Dato un intero positivo t, supponiamo di modificare i profitti del nostro problema nel modo seguente

$$\bar{v}_i = \left\lfloor \frac{v_i}{10^t} \right\rfloor \times 10^t,$$

il che equivale ad azzerare le ultime t cifre del profitto. Ad esempio, per $t = 2$ e $v_i = 3453$ abbiamo $\bar{v}_i = 3400$. Ovviamente, possiamo risolvere il problema con i nuovi profitti $\bar{v}_i$, dividendoli tutti per 10^t e moltiplicando il valore ottimo alla fine per 10^t.

Osservazione 2 *Risolvere il problema modificato richiede un numero di operazioni pari al più a $O(n^2 v_{max} 10^{-t})$ dove v_{max} denota il massimo tra i profitti degli n oggetti.*

Dimostrazione Notando che il valore ottimo del problema ottenuto dividendo i profitti $\bar{v}_i$ per 10^t non può essere superiore a $10^{-t}v^*$, l'Osservazione 1 ci dice che risolvere il problema modificato con la procedura vista in precedenza richiede un numero di operazioni pari al più a $O(n10^{-t}v^*)$. Indicando con v_{max} il massimo tra i profitti degli oggetti, possiamo anche scrivere che il numero di operazioni è $O(n^2 v_{max} 10^{-t})$ (ovviamente si ha $v^* \leq nv_{max}$).

Sia ora N' la soluzione del problema modificato e N^* quella del problema originario. Introduciamo inoltre una terza soluzione

$$N'' = \begin{cases} N' & \text{se } \sum_{i \in N'} v_i \geq v_{max} \\ \{i_{max}\} & \text{altrimenti} \end{cases}$$

dove i_{max} è l'oggetto con il profitto massimo v_{max}. In pratica, N'' coincide con N' a meno che gli oggetti in N' abbiano un valore complessivo inferiore a quello dell'oggetto con profitto massimo, nel qual caso si utilizza al posto di N' la soluzione costituita dal solo oggetto di profitto massimo. Rispetto al calcolo di N', il calcolo di N'' richiede un'ulteriore somma di al più n addendi, ma questa non modifica l'ordine di grandezza del numero di operazioni individuato nell'Osservazione 2. Si dimostra il seguente risultato.

Osservazione 3 *Si ha che*

$$\frac{\sum_{i \in N^*} v_i}{\sum_{i \in N''} v_i} \leq 1 + \frac{n10^t}{v_{max}},$$

Dimostrazione Calcoliamo un bound dal di sopra per $\sum_{i \in N^*} v_i - \sum_{i \in N'} v_i$. Abbiamo:

$$\sum_{i \in N^*} v_i \geq \sum_{i \in N'} v_i \geq \sum_{i \in N'} \bar{v}_i \geq \sum_{i \in N^*} \bar{v}_i \geq \sum_{i \in N^*} (v_i - 10^t) \geq \sum_{i \in N^*} v_i - n10^t,$$

da cui

$$\sum_{i \in N^*} v_i - \sum_{i \in N'} v_i \leq n10^t.$$

Dal momento che $\sum_{i \in N''} v_i \geq \sum_{i \in N'} v_i$ abbiamo anche:

$$\sum_{i \in N^*} v_i - \sum_{i \in N''} v_i \leq n10^t.$$

Se si dividono entrambi i membri per $\sum_{i \in N''} v_i$ abbiamo:

$$\frac{\sum_{i \in N^*} v_i}{\sum_{i \in N''} v_i} \leq 1 + \frac{n10^t}{\sum_{i \in N''} v_i}.$$

da cui, tenendo conto che si ha anche $\sum_{i \in N''} v_i \geq v_{max}$, si arriva a:

$$\frac{\sum_{i \in N^*} v_i}{\sum_{i \in N''} v_i} \leq 1 + \frac{n10^t}{v_{max}}.$$

come si voleva dimostrare.

Si prenda ora

$$t = \left\lfloor \log_{10} \left(\frac{\varepsilon v_{max}}{n} \right) \right\rfloor.$$

Si noti che in tal caso, in base all'Osservazione 2, il tempo di esecuzione della procedura per risolvere il problema modificato è $O\left(\frac{n^3}{\varepsilon}\right)$, ovvero è polinomiale sia rispetto al numero di oggetti (e quindi rispetto alla dimensione del problema) sia rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. Inoltre, per la definizione di t si ha

$$\frac{n10^t}{v_{max}} \leq \varepsilon$$

e quindi, in base all'Osservazione 3, abbiamo anche

$$\frac{\sum_{i \in N^*} v_i}{\sum_{i \in N''} v_i} \leq 1 + \varepsilon.$$

Possiamo quindi concludere che la procedura che ci restituisce N'' rappresenta un FPTAS per il problema `KNAPSACK`. Teniamo presente che quello descritto è un possibile FPTAS per `KNAPSACK`. Esiste anche un'altra tecnica, detta di *scaling-rounding*, con complessità $O\left(\frac{n}{\varepsilon^2}\right)$.

Capitolo 15

Tecniche Euristiche

In questi appunti ci siamo limitati a considerare algoritmi esatti o di approssimazione per problemi di ottimizzazione. Nella pratica accade di frequente che per varie ragioni i tempi accettabili per le istanze di un problema da risolvere in determinate applicazioni, siano di gran lunga inferiori a quelli che possono garantire algoritmi esatti o di approssimazione. Questo può accadere quando dobbiamo risolvere un'istanza di grandi dimensioni di un problema difficile (per esempio un'istanza di `TSP`) ma anche se dobbiamo risolvere istanze di problemi risolvibili in tempo polinomiale ma con tempi accettabili molto piccoli (si pensi al caso di applicazioni real-time). In questi casi non si rinuncia a risolvere i problemi ma si rinuncia a certificare la qualità della soluzione restituita. In tal caso infatti si utilizzano le *tecniche euristiche* che cercano di:

- avere tempi di esecuzione che non crescano troppo rapidamente rispetto alla dimensione del problema (polinomiali con esponente non troppo elevato);

- restituire soluzioni che siano almeno per molte istanze del problema ottime o comunque vicine a quelle ottime.

Un'euristica è quindi un compromesso tra due esigenze contrastanti: la qualità della soluzione ottenuta (che dovrebbe essere il più vicino possibile al valore ottimo) e la rapidità con cui viene restituita una soluzione (non necessariamente ottima). Nelle euristiche si rinuncia alla garanzia di ottenere *sempre* una soluzione ottima (cosa che richiederebbe tempi troppo elevati) per poter ottenere una risposta in tempi accettabili. Nel seguito vedremo molto rapidamente alcuni esempi di euristiche.

15.1 Alcuni approcci euristici

15.1.1 Algoritmo greedy

In un algoritmo greedy la soluzione viene costruita un passo per volta facendo ad ogni passo una scelta greedy (golosa) ovvero una scelta che non è necessariamente la migliore in assoluto ma è la migliore in quel dato momento. Un esempio di tecnica greedy è già stato visto nella Sezione 10.1.3 dove abbiamo introdotto un algoritmo greedy per il problema `MST`. In quel caso la scelta greedy consisteva nel prendere in esame ad ogni iterazione l'arco con peso minimo tra tutti quelli non ancora analizzati. Per tale algoritmo greedy è improprio parlare di euristica. Sappiamo infatti che esso risolve in modo *esatto* il problema `MST`. Ma vediamo ora un altro esempio in cui siamo meno fortunati e l'algoritmo greedy non restituisce sempre una soluzione ottima. Sia dato un problema `TSP` che per semplicità supporremo su di un grafo completo $G = (V, A)$. Un algoritmo greedy per tale problema è il seguente.

Passo 1 Si fissi un nodo di partenza i e lo si inserisca in un insieme W, ovvero $W = \{i\}$. Si ponga $r = i$.

Passo 2 Si consideri il nodo $s \in V \setminus W$ con distanza minima dal nodo r, cioè

$$d_{rs} = \min_{j \in V \setminus W} d_{rj},$$

dove d_{ij} denota la distanza da percorrere lungo l'arco (i, j). Si faccia seguire r da s nel circuito.

Passo 3 Si ponga $W = W \cup \{s\}$ e $r = s$. Se $W = V$ ci si arresta chiudendo il circuito andando da s al nodo iniziale i. Altrimenti si ritorni al Passo 2.

In pratica ad ogni iterazione ci si muove dall'ultimo nodo raggiunto, il nodo r, al nodo s che è il più vicino (scelta greedy) a r tra tutti quelli non ancora visitati dal circuito. Si può dimostrare che il numero di operazioni di tale algoritmo è dell'ordine di n^2, dove $n =\mid V \mid$. Si tratta quindi di un algoritmo con complessità polinomiale. Vediamo ora di applicare l'algoritmo sull'esempio in Figura 15.1 dove $M > 0$. Partendo dal nodo 1 l'algoritmo si sposta verso il nodo 3. Da questo si muove verso il nodo 2, dal nodo 2 si va verso il nodo 4 ed infine dal nodo 4 si chiude il circuito ritornando al nodo 1. L'algoritmo restituisce quindi il circuito C_1

$$1 \to 3 \to 2 \to 4 \to 1$$

con $f(C_1) = 1 + 1 + M + 2 = 4 + M$. Questa è la soluzione ottima solo per $M \leq 3$ ma per $M > 3$ la soulzione ottima è il circuito C_2

$$1 \to 2 \to 3 \to 4 \to 1$$

con $f(C_2) = 2 + 1 + 2 + 2 = 7$. Si può anche notare che per $M > 3$

$$\frac{f(C_1)}{f(C_2)} = \frac{4 + M}{7}.$$

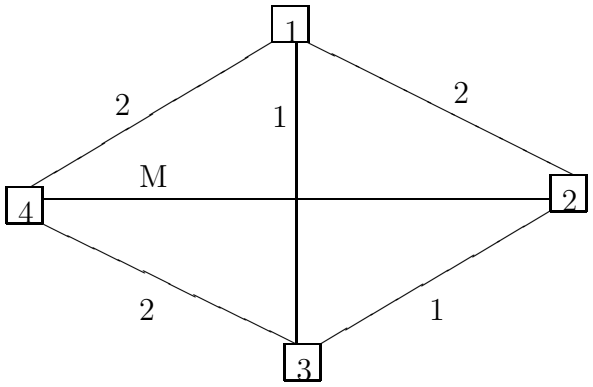


Figura 15.1: Un grafo G su cui applicare l'algoritmo greedy per determinare il circuito hamiltoniano a distanza minima.

Al crescere di M il rapporto cresce all'infinito, mostrando quindi come già tra queste piccole istanze sia sempre possibile trovarne alcune per cui questa euristica non è in grado di risolvere il problema di r-approssimazione per ogni $r > 0$. Questo non è altro che una conferma di quanto già visto in precedenza: per il problema `TSP` è improbabile che esista una procedura di risoluzione polinomiale che risolva il problema di r-approssimazione per *ogni* istanza di tale problema e per ogni $r > 0$.

15.1.2 Ricerca locale

Per poter definire un algoritmo di ricerca locale dobbiamo introdurre il concetto di vicinanza per un elemento della regione ammissibile S.

Definizione 11 *Una* vicinanza N *nella regione ammissibile* S *è definita come una funzione*

$$N : S \to 2^S$$

che ad ogni elemento $x \in S$ *associa un sottinsieme di punti di* S *detti* vicini *di* x.

Dato un problema di ottimizzazione combinatoria non esiste necessariamente un'unica vicinanza per esso.

Esempio 49 *Si consideri il problema* `TSP`. *Dato un circuito hamiltoniano* C *la vicinanza* $N_k(C)$ *con* $k \leq n$ *è costituita da tutti i circuiti hamiltoniani ottenibili rimuovendo* k *archi da* C *ed aggiungendo altri* k *archi (non necessariamente diversi dai precedenti). Per ogni* $k = 2, 3, \ldots, n$ *si ha una diversa vicinanza. Nell'esempio di Figura 15.2 se considero il circuito* C

$$1 \to 2 \to 3 \to 4 \to 5 \to 1$$

la vicinanza $N_2(C)$ è costituita dai 6 circuiti hamiltoniani

$$1 \to 2 \to 3 \to 4 \to 5 \to 1$$

$$1 \to 3 \to 2 \to 4 \to 5 \to 1$$

$$1 \to 4 \to 3 \to 2 \to 5 \to 1$$

$$1 \to 2 \to 4 \to 3 \to 5 \to 1$$

$$1 \to 2 \to 5 \to 4 \to 3 \to 1$$

$$1 \to 2 \to 3 \to 5 \to 4 \to 1$$

Per $k = 5$ si avrà invece che $N_5(C)$ comprende tutti i 12 circuiti hamiltoniani.

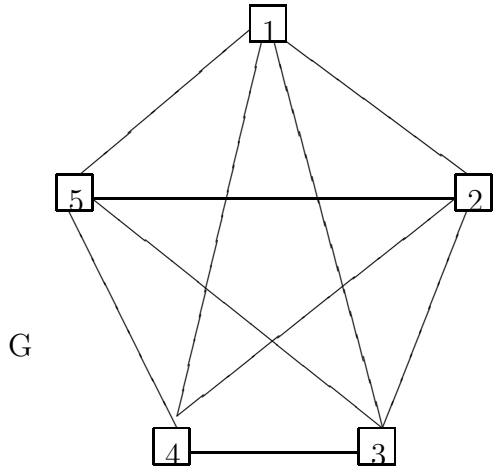


Figura 15.2: Un grafo completo G con 5 nodi.

Una volta fissata una vicinanza N è possibile introdurre la definizione di ottimo locale (limitata ad un problema di minimo ma facilmente estendibile a problemi di massimo).

Definizione 12 *Un elemento $\overline{x} \in S$ si definisce ottimo locale (rispetto alla vicinanza N) se*

$$\forall\ y \in N(\overline{x}) \quad f(y) \geq f(\overline{x}),$$

cioè un punto è un ottimo (minimo) locale se ha valore della funzione obiettivo f non peggiore (non superiore) rispetto a tutti i suoi vicini.

Si noti che un punto di ottimo globale è sempre anche un ottimo locale indipendentemente dalla vicinanza. Non è invece sempre vero il viceversa. Una vicinanza in cui ogni ottimo locale è anche ottimo globale viene detta *vicinanza esatta*. Nel problema `TSP` la vicinanza N_n (in cui ogni circuito hamiltoniano ha come vicini tutti i circuiti hamiltoniani) è esatta, mentre la vicinanza N_2 non è esatta (vi sono ottimi locali che non sono ottimi globali).

Esempio 50 *Come ulteriore esempio esaminiamo il problema illustrato in Figura 15.3 dove S è la griglia di punti*

$$\{(i,j):\ i=1,\ldots,5\ \ j=1,\ldots,5\}$$

e la funzione obiettivo (i cui valori sono riporati all'interno dei nodi della griglia) è data da

$$f(i,j)=25(i-2)^2(i-4)^2+i+25(j-2)^2(j-4)^2+j. \qquad (15.1)$$

La struttura di vicinanza è definita nel modo seguente:

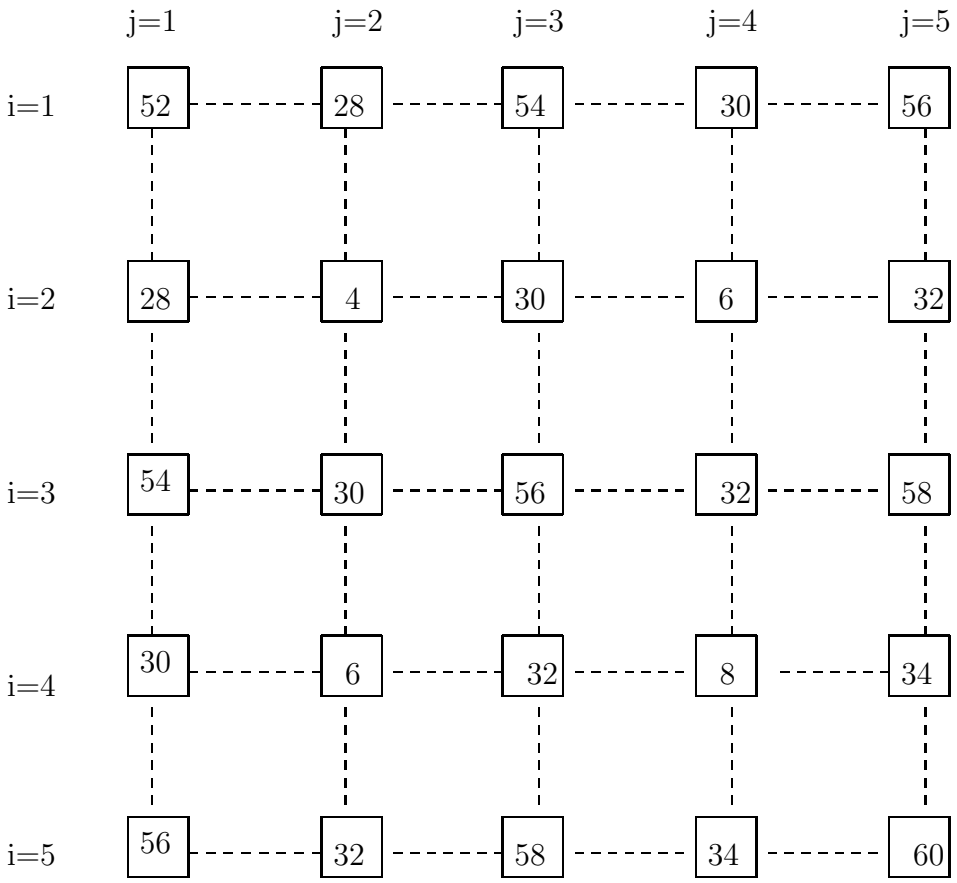


Figura 15.3: Un problema di ottimizzazione combinatoria con una possibile struttura di vicinanza.

$$\begin{aligned} N(i,j) \quad = \quad & \{(i,h)\in S:\ h=j-1,j,j+1\}\cup \\ & \cup\{(h,j)\in S:\ h=i-1,i,i+1\} \end{aligned}$$

(nella figura ogni punto (i,j) è collegato ai suoi vicini tramite un arco). Si noti che l'ottimo globale è il punto $(2,2)$ ma vi sono anche tre ottimi locali non globali: $(2,4),(4,2)$ e $(4,4)$. La vicinanza quindi non è esatta. Se definiamo un'altra vicinanza

$$\begin{aligned} N'(i,j) &= \{(i,h) \in S:\ h = j-2, j-1, j, j+1, j+2\} \cup \\ &\quad \cup \{(h,j) \in S:\ h = i-2, i-1, i, i+1, i+2\} \end{aligned}$$

(si noti che $N'(i,j) \supset N(i,j)$ per ogni coppia (i,j)), abbiamo invece un unico ottimo locale e globale, il punto $(2,2)$, e quindi la vicinanza è esatta.

Dato un problema di ottimizzazione combinatoria (f,S) ed una vicinanza N per esso, possiamo introdurre un algoritmo di ricerca locale.

Passo 1 Sia $x_0 \in S$ e si ponga $k = 0$.

Passo 2 Se per ogni $y \in N(x_k)$ si ha $f(y) \geq f(x_k)$, ovvero se x_k è un ottimo locale, allora ci arrestiamo restituendo x_k. Altrimenti andiamo al Passo 3.

Passo 3 Si selezioni $\overline{y} \in N(x_k)$ tale che $f(\overline{y}) < f(x_k)$ e si ponga $x_{k+1} = \overline{y}$.

Passo 4 Si ponga $k = k+1$ e si ritorni al Passo 2.

Per esempio nel problema di Figura 15.3 posso partire con $x_0 = (5,5)$. Da qui posso spostarmi in $x_1 = (4,5)$ e da qui in $x_2 = (4,4)$. Essendo x_2 un ottimo locale mi arresto e resituisco il punto x_2. Se la vicinanza è esatta, l'algoritmo di ricerca locale mi restituisce l'ottimo globale. Se non è esatta può succedere che mi venga restituito un ottimo locale ma non globale. Nel nostro esempio è esattamente ciò che accade: ci viene restituito il punto $x_2 = (4,4)$ che è un ottimo locale ma non globale. Da quanto detto sembrerebbe auspicabile utilizzare sempre vicinanze esatte. Ad esempio, nel problema `TSP` sembrerebbe auspicabile utilizzare la vicinanza esatta N_n piuttosto che quella non esatta N_2. Ma non dobbiamo dimenticare che a noi non interessa soltanto il risultato di un algoritmo ma anche il tempo necessario per la sua esecuzione. Nel caso degli algoritmi di ricerca locale il tempo di esecuzione è strettamente legato ai tempi necessari per esplorare i vicini del punto corrente ai Passi 2 e 3 dell'algoritmo. Nei problemi più difficili le vicinanze esatte hanno anche dimensioni molto elevate e la loro esplorazione richiede tempi molto elevati. Ad esempio, la vicinanza N_n del problema `TSP` è tale che ogni circuito abbia come vicini tutti i circuiti hamiltoniani e quindi l'esplorazione dei vicini di un circuito si riduce all'enumerazione completa di tutti i circuiti che, come già visto, è eseguibile in tempi ragionevoli solo per istanze molto piccole. Molto minore è il numero di vicini se si usa la vicinanza N_2 (tale numero è dell'ordine di n^2; più in generale tale numero è $O(n^k)$ per la vicinanza N_k). D'altra parte usando la vicinanza N_2 non si ha alcuna garanzia che la ricerca locale restituisca un ottimo globale. La scelta di una vicinanza dovrà basarsi su un compromesso tra tempi di esecuzione e qualità della soluzione restituita dalla ricerca locale.

Studi sperimentali indicano che la vicinanza N_3 fornisce risultati molto migliori rispetto alla N_2, seppure in tempi più elevati, mentre i risultati ottenuti con la vicinanza N_4 non sono di molto migliori rispetto a quelli ottenibili con la N_3 nonostante i più elevati tempi di esecuzione.

15.1.3 Simulated Annealing

Gli algoritmi Simulated Annealing (SA nel seguito) sono basati su un'analogia con un fenomeno fisico: mentre a temperature elevate le molecole in un liquido tendono a muoversi liberamente, se la temperatura viene decrementata in modo sufficientemente lento, la mobilità termica delle molecole viene persa e tendono a formare un cristallo puro che corrisponde anche ad uno stato di minima energia. Se la temperatura viene decrementata troppo velocemente si parla di *quenching* e lo stato finale è uno stato policristallino od amorfo con un'energia più elevata di quella del cristallo puro. Stabilendo un'analogia tra configurazioni di un sistema fisico ed elementi della regione ammissibile da un lato e tra energia del sistema e funzione obiettivo dall'altro, ovvero trattando il problema di ottimizzazione come un particolare sistema fisico, si è arrivati a formulare gli algoritmi SA per problemi di ottimizzazione combinatoria attraverso la simulazione del processo fisico. Come nel processo fisico una lenta decrescita della temperatura conduce allo stato di minima energia, allo stesso modo nel problema di ottimizzazione combinatoria vogliamo arrivare in un punto (l'ottimo globale) con valore minimo della funzione obiettivo. La struttura degli algoritmi SA è la seguente.

Passo 1 Sia $x_0 \in S$ e si fissi $k = 0$.

Passo 2 Si generi in modo casuale un punto $y_{k+1} \in N(x_k)$.

Passo 3 Si ponga $x_{k+1} = y_{k+1}$ con probabilità pari a

$$\min\left\{\exp\left\{\frac{f(x_k) - f(y_{k+1})}{T_k}\right\}, 1\right\}, \tag{15.2}$$

dove T_k è un parametro non negativo che, per analogia con il fenomeno fisico, viene chiamato temperatura. Altrimenti si ponga $x_{k+1} = x_k$. Nel primo caso si dice che il punto y_{k+1} viene *accettato* come nuovo punto, nel secondo che viene *rigettato*.

Passo 4 (Cooling Schedule) Si aggiorni il valore T_{k+1} della temperatura.

Passo 5 Si verifichi un criterio di arresto (ad esempio ci si arresti se il numero k di iterazioni è superiore ad un numero prefissato MAX_ITER di iterazioni). Se il criterio è soddisfatto ci si arresti e si restituisca il miglior elemento di S osservato durante l'esecuzione dell'algoritmo. Se invece non è soddisfatto si ponga $k = k + 1$ e si ritorni al Passo 2.

La probabilità (15.2) viene detta funzione di accettazione di Metropolis. In essa si può notare che ci si sposta sempre in un punto y_{k+1} se questo ha un valore di f inferiore rispetto al punto corrente x_k, esattamente come avviene in una ricerca

locale. Ma la novità rispetto alla ricerca locale è che ci si può spostare anche in punti peggiori con una probabilità che è tanto più bassa quanto peggiore (più elevato) è il valore di f nel punto y_{k+1} rispetto al valore di f nel punto corrente x_k. Si noti inoltre che tale probabilità è controllata dal parametro temperatura: per valori elevati della temperatura è molto probabile accettare anche punti di molto peggiori, ma man mano che si decresce la temperatura la probabilità di accettare un punto peggiore diventa sempre minore. La ragione per cui si accettano anche punti con valori della funzione obiettivo peggiori è che questo è il solo modo di sfuggire ad ottimi locali non globali. Nella ricerca locale applicata all'esempio in Figura 15.3 abbiamo visto come una volta giunti nell'ottimo locale x_2 non siamo più in grado di procedere e siamo costretti ad arrestarci (siamo intrappolati nell'ottimo locale). In un algoritmo SA possiamo invece sfuggire da un ottimo locale accettando anche punti peggiori rispetto a quello corrente. Per questa ragione tali algoritmi vengono detti *hill-climbing*, scavalcano le colline che separano tra loro gli ottimi locali.
Una componente essenziale degli algoritmi SA è il Passo 4 detto di *cooling schedule*, cioè il passo in cui si aggiorna la temperatura. Tipicamente si inizia con una temperatura elevata e di seguito la temperatura viene diminuita nel corso delle iterazioni dell'algoritmo. Se potessimo eseguire l'algoritmo per un tempo infinito allora si può dimostrare che con una probabiltà pari a 1 esso sarebbe in grado di individuare l'ottimo globale, *a patto di far decrescere la temperatura in modo sufficientemente lento*, proprio come avviene nel fenomeno fisico. In particolare si deve avere ad ogni iterazione k che

$$T_k \geq \frac{M}{\log(k)},$$

dove M è una costante dipendente dal problema. Una decrescita più rapida (il quenching del fenomeno fisico) può fare in modo che si rimanga intrappolati in un ottimo locale ma non globale. In realtà sappiamo bene che non si ha mai a disposizione un tempo infinito e neppure molto elevato. Quindi ad una scelta che garantirebbe l'individuazione dell'ottimo globale ma in tempi troppo elevati, si preferisce spesso una scelta di temperature che decrescono più rapidamente. Tale scelta non esclude di rimanere intrappolati in ottimi locali, ma consente di giungere in tempi più brevi ad una buona soluzione. Non esploreremo oltre la questione delle temperature, precisando però che scelte come quale *temperatura iniziale* utilizzare, *quando* ridurre la temperatura e *di quanto* ridurla, sono molto delicate per il buon funzionamento dell'algoritmo e sono tipicamente legate al problema da risolvere.

15.1.4 Algoritmi genetici

Diamo ora una breve descrizione di un altro approccio euristico, gli algoritmi genetici. Anche questi algoritmi nascono da un'analogia, l'analogia con il concetto darwiniano di *selezione naturale*. Ad una data iterazione k tali algoritmi lavorano su una popolazione P_k di elementi della regione ammissibile S. Da tale popolazione vengono generati nuovi elementi di S attraverso i processi di

mutazione, che genera nuovi elementi di S modificando singoli elementi della popolazione P_k, e di *ricombinazione*, che forma nuovi elementi di S mettendo assieme pezzi di coppie di elementi di P_k (più in generale la ricombinazione può riguardare non solo coppie di elementi di P_k ma anche più di due elementi della popolazione; qui ci restringiamo al caso, molto frequente, di ricombinazione di due elementi). Indicando con O_k i nuovi elementi di S generati tramite mutazione e ricombinazione, abbiamo una popolazione allargata $P_k \cup O_k$. A questo punto interviene il processo di *selezione*: all'iterazione successiva arriverà un sottinsieme P_{k+1} di $P_k \cup O_k$, ovvero solo alcuni degli elementi in $P_k \cup O_k$ sopravviveranno e faranno parte della popolazione P_{k+1} all'iterazione $k+1$. Nella scelta delle coppie da ricombinare ed anche nella selezione si tendono a favorire elementi della popolazione con un elevato valore di *fitness* (adattamento). Come ovvia (ma non necessariamente unica) misura di fitness si può pensare al valore della funzione obiettivo f: tanto più piccolo è il valore di f per un elemento in un problema di minimo (tanto più grande in un problema di massimo), qunato maggiore è il valore di fitness di tale elemento. Siamo ora pronti a fornire lo schema di un generico algoritmo genetico.

Passo 1 Si generi una popolazione $P_0 \subset S$ e si ponga $k = 0$.

Passo 2 (mutazione) Si scelgano alcuni elementi in P_k e si generino tramite mutazione nuovi elementi di S. Si inseriscano tali nuovi elementi in un insieme O_k.

Passo 3 (ricombinazione) Si scelgano coppie di elementi in P_k (tipicamente la scelta avviene in modo casuale ma favorendo elementi con un miglior valore di fitness) e si ricombinino tali coppie generando nuovi elementi di S. Si aggiungano tali nuovi elementi in O_k.

Passo 4 (selezione) Si determini P_{k+1} selezionando alcuni degli elementi in $P_k \cup O_k$ (tipicamente in P_{k+1} vengono conservati, tra gli altri, anche un certo numero di elementi con il miglior valore di fitness tra tutti quelli in $P_k \cup O_k$).

Passo 5 Si verifichi un criterio di arresto. Se è soddisfatto ci si arresti restituendo la miglior soluzione trovata durante l'esecuzione dell'algoritmo. Altrimenti si ponga $k = k + 1$ e si ritorni al Passo 2.

Per illustrare brevemente i processi di mutazione e ricombinazione, possiamo considerare il problema illustrato in Figura 15.3. Per tale problema il processo di mutazione può essere considerato semplicemente come la generazione di un vicino dell'elemento su cui agisce la mutazione. Quindi una mutazione che agisce sull'elemento $(4, 5)$ può generare, ad esempio, l'elemento $(4, 4)$. Più significativa è la ricombinazione. Supponiamo di avere una coppia di elementi (i, j) e (h, k). Il processo di ricombinazione può agire generando a partire da tale coppia di elementi, due nuovi elementi: uno con la prima componente del primo elemento e la seconda del secondo, quindi (i, k) e l'altro con la prima componente del secondo elemento e la seconda del primo, quindi (h, j). Per

esempio, se la coppia di elementi è $(2, 5)$ e $(5, 2)$, i due nuovi elementi saranno $(2, 2)$ e $(5, 5)$. È importante notare che mentre i due elementi $(2, 5)$ e $(5, 2)$ sono lontani dall'ottimo globale, uno dei loro figli, è l'ottimo globale stesso $(2, 2)$. Quindi, mentre nelle ricerche locali, negli algoritmi SA e nella mutazione stessa degli algoritmi genetici ci possiamo solo spostare in punti vicini, con la ricombinazione possiamo anche balzare in un solo colpo in una zona molto lontana da entrambi gli elementi della coppia. In certi casi il processo di ricombinazione fornisce dei grossi vantaggi. Questo succede in particolare quando il problema gode di una certa separabilità. Non si vuole qui approfondire cosa si intenda per separabilità ma ne possiamo dare una nozione intuitiva attraverso il problema di Figura 15.3. Se ne osserviamo la funzione obiettivo (15.1), si nota che essa è formata da due termini, $(i-2)^2(i-4)^2+i$, dove compare la sola prima componente degli elementi (i, j) di S, e $(j-2)^2(j-4)^2+j$, dove compare la sola seconda componente. In questo caso quindi intendiamo per separabilità il fatto che la funzione obiettivo può essere scomposta in due parti dove le componenti che formano gli elementi di S compaiono separatamente. Se osserviamo i due elementi $(2, 5)$ e $(5, 2)$ si può notare che il primo elemento ha la prima componente ottima rispetto al primo termine $(i-2)^2(i-4)^2+i$ ma non rispetto al secondo termine $(j-2)^2(j-4)^2+j$, mentre per il secondo elemento vale esattamente il viceversa. A questo punto la ricombinazione, mettendo assieme il pezzo buono del primo elemento (la prima componente) ed il pezzo buono del secondo (la seconda componente) è in grado di generare un nuovo elemento migliore di entrambi (in questo caso addirittura l'ottimo globale).

Capitolo 16

Ottimizzazione non lineare

In questo capitolo ci occuperemo di problemi di ottimizzazione non lineare. La forma generica di un problema di ottimizzazione non lineare è la seguente:

$$\begin{array}{ll} \min & f(x) \\ & x \in \mathbb{R}^n \\ & c_i(x) = 0 \quad i \in K_1 \\ & c_i(x) \geq 0 \quad i \in K_2 \end{array} \tag{16.1}$$

con funzione obiettivo f e regione ammissibile

$$S = \left\{ \begin{array}{ll} x \in \mathbb{R}^n & \\ c_i(x) = 0 & i \in K_1 \\ c_i(x) \geq 0 & i \in K_2 \end{array} \right\}$$

(teniamo presente che i problemi di massimizzazione possono sempre essere ricondotti a problemi di minimizzazione con un cambio di segno nell'obiettivo: $\max f = -\min -f$). Supporremo che:

- $f, c_i \in \mathcal{C}^2$ (ipotesi di differenziabilità);
- almeno una tra le funzioni $f, c_i,\ i \in K_1 \cup K_2$ è *non* lineare (ipotesi di nonlinearità).

Il caso in cui f e $c_i,\ i \in K_1 \cup K_2$ sono tutte funzioni lineari corrisponde alla classe dei problemi di PL, già trattati in precedenza. Diamo ora le seguenti definizioni.

Definizione 13 Minimo globale: *un punto $x^* \in S$ tale che:*

$$f(x^*) \leq f(x) \quad \forall\ x \in S.$$

Minimo locale forte*: un punto x^* tale che per qualche $\delta > 0$:*

$$f(x^*) < f(x) \quad \forall\ x \in S \cap \{x\ :\ \|x - x^*\|_2 \leq \delta\},\ x \neq x^*.$$

(con $\|\cdots\|_2$ si indica la norma euclidea).
Minimo locale debole*: un punto x^* tale che per qualche $\delta > 0$:*

$$f(x^*) \leq f(x) \quad \forall\, x \in S \cap \{x \;:\; \|x - x^*\|_2 \leq \delta\}$$

e x^* non è un minimo locale forte.

Abbiamo diversi casi possibili.

- non esitono minimi locali e/o globali perché la funzione è illimitata inferiormente. Esempio: $f(x) = x^3$, $S = \mathbb{R}$;
- esitono minimi locali ma non minimi globali perché la funzione è illimitata inferiormente. Esempio: $f(x) = x^3 - x$, $S = \mathbb{R}$ dove $x' = 1/\sqrt{3}$ è minimo locale ma non globale.
- non esitono minimi locali e/o globali anche se la funzione è limitata inferiormente. Esempio: $f(x) = e^{-x}$, $S = \mathbb{R}$
- un minimo globale (e quindi anche locale) esiste certamente se:
  - S è compatto (chiuso e limitato);
  - f continua.

  (teorema di Weierstrass).

Che cosa possiamo dire, in termini di teoria della complessità, per quanto concerne l'identificazione di ottimi globali e/o locali? Introduciamo il seguente problema detto `SUBSET SUM`: dati gli interi positivi $d_0, d_1, \ldots, d_n$, ci chiediamo se esiste una soluzione del sistema

$$\begin{array}{ll} \sum_{j=1}^{n} d_j y_j = d_0 & \\ y_j \in \{0,1\} & j = 1, \ldots, n \end{array}$$

In altre parole, si cerca di stabilire se esiste un sottinsieme degli interi $d_1, \ldots, d_n$ la cui somma è pari a d_0. Si può dimostrare che tale problema è $\mathcal{NP}$-completo. Ora, dai dati del problema `SUBSET SUM` deriviamo il seguente problema con funzione obiettivo quadratica e vincoli box:

$$\begin{array}{ll} \min & \left(\sum_{j=1}^{n} d_j y_j - d_0\right)^2 + \sum_{j=1}^{n} y_j(1 - y_j) \\ & 0 \leq y_j \leq 1 \qquad\qquad j = 1, \ldots, n \end{array}$$

Si ha che l'ottimo globale di questo problema ha valore dell'obiettivo pari a 0 se e solo se il corrispondente problema `SUBSET SUM`.

Quindi, calcolare il minimo globale di un problema non convesso è un problema difficile anche nel caso di funzioni quadratiche e vincoli semplici, come i vincoli box.

Se la difficoltà di individuare minimi globali era prevedibile, un po' meno lo è quella dei seguenti problemi:

- nei problemi con e senza vincoli, stabilire se un dato punto *non* è un punto di minimo locale è $\mathcal{NP}$-completo;
- nei problemi con e senza vincoli, stabilire se l'obiettivo del problema è illimitato sulla regione ammissibile è NP-completo.

Le cose vanno un po' meglio quando si impongono restrizioni sulle funzioni f e c_i. Abbiamo innanzitutto bisogno di una definizione.

Definizione 14 *Data una matrice simmetrica A di ordine n, diciamo che questa è* semidefinita positiva *se:*

$$x^T Ax \geq 0 \qquad \forall\ x \in \mathbb{R}^n.$$

Diciamo che questa è definita positiva *se:*

$$x^T Ax > 0 \qquad \forall\ x \in \mathbb{R}^n \setminus \{0\}.$$

Vale la seguente osservazione.

Osservazione 46 *Una matrice A è semidefinita (definita) positiva se e solo se tutti i suoi autovalori sono non negativi (positivi).*

Ricordiamo che data una matrice A, i suoi autovalori sono le radici della seguente equazione:

$$det(A - \lambda I) = 0$$

dove I è la matrice identica. Diamo ora la seguente definizione.

Definizione 15 *Una funzione f si dice* convessa *se (condizioni equivalenti):*

- $$\forall\ x_1, x_2 \in \mathbb{R}^n,\ \forall\ \lambda \in (0,1)\ :\ f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2);$$
- $$\forall\ x_1, x_2 \in \mathbb{R}^n\ :\ f(x_2) \geq f(x_1) + \nabla f(x_1)^T (x_2 - x_1);$$
  dove $\nabla f(x_1) = \left(\frac{\partial f}{\partial x_j}(x_1)\right)_{j=1,\ldots,n}$ *indica il gradiente della f in x_1;*
- $$\forall\ x_1 \in \mathbb{R}^n\ :\ \nabla^2 f(x_1)\ \text{ è semidefinita positiva.}$$
  dove $\nabla^2 f(x_1) = \left(\frac{\partial^2 f}{\partial x_i\ x_j}(x_1)\right)_{i,j=1,\ldots,n}$ *indica l'Hessiano della f in x_1.*

Nel caso in cui le diseguaglianze siano strette e l'Hessiano sia definito positivo, si parla di funzione strettamente convessa. *Definiamo una funzione f* concava *(*strettamente concava*) se $-f$ è convessa (strettamente convessa).*

Passiamo quindi a definire la classe dei problemi di Programmazione Convessa (PC).

Definizione 16 *I problemi di* Programmazione Convessa (PC) *hanno la seguente forma*

$$\begin{array}{ll} \min & f(x) \\ & x \in \mathbb{R}^n \\ & c_i(x) \geq 0 \quad i \in K_2 \end{array}$$

con f convessa e le c_i, $i \in K_2$, concave.

Si noti che i problemi di PL sono una sottoclasse dei problemi di PC. Vale la seguente osservazione.

Osservazione 47 *I problemi PC appartengono alla classe $\mathcal{P}$. Gli algoritmi di complessità polinomiale che hanno consentito di stabilire questo sono gli stessi (elissoide, punto interno) che hanno permesso di catalogare nella classe $\mathcal{P}$ i problemi di PL.*

Nell'ambito del corso non potremo effettuare una trattazione completa di tutti gli aspetti dei problemi di ottimizzazione non lineare Per questa ragione, nel seguito concentreremo la nostra attenzione sul problema dell'individuazione di minimi locali e, dal punto di vista algoritmico, ci restringeremo ulteriormente ai problemi non vincolati. Questo chiaramente rappresenta una restrizione rispetto ai problemi generali introdotti sopra, sia perché non si discute di possibili approcci (esatti o euristici) per l'individuazione di minimi globali, sia perché si tralascia la trattazione dei vincoli da un punto di vista algoritmico (per eventuali approfondimenti su tali argomenti si rimanda a testi specializzati). Tuttavia già la trattazione degli argomenti citati rappresenta un'utile introduzione alle problematiche dell'ottimizzazione non lineare, che possono poi venire utili anche per la trattazione dei casi non trattati in questi appunti.

16.1 Il caso non vincolato

Nel caso non vincolato gli insiemi K_1 e K_2 in (16.1) devono essere entrambi vuoti. Vedremo per prima cosa condizioni necessarie e sufficienti di ottimalità locale per questo caso (condizioni di ottimalità per il caso vincolato verranno presentate più avanti) . In seguito, passeremo alla descrizione di alcuni aprrocci risolutivi.

16.1.1 Condizioni di ottimalità

Dato un generico problema di ottimizzazione non lineare (16.1), non è possibile la verifica di ottimalità *locale* di un punto sulla base della Definizione 13. Si possono però introdurre delle *condizioni di ottimalità locale* che, come già detto, restringeremo al caso non vincolato ($K_1 = K_2 = \emptyset$).

Prima di prendere in esame le condizioni di ottimalità, richiamiamo qui alcuni risultati sui gradienti di funzioni che verranno usati più avanti. Data

$f : \mathbb{R}^n \to \mathbb{R}$, si ha che

$$\nabla f(x) = \begin{cases} c & \text{se } f(x) = c^T x + c_0 \\ Hx & \text{se } f(x) = \frac{1}{2} x^T H x, \quad H \text{ simmetrica} \end{cases}$$

Dati $x_i : \mathbb{R} \to \mathbb{R}^{n_i}$, $i = 1, \ldots, k$, e $f : \mathbb{R}^{\sum_{i=1}^{k} n_i} \to \mathbb{R}$, definiamo

$$g(\lambda) = f(x_1(\lambda), \ldots, x_k(\lambda)).$$

Per le regole delle derivate di funzioni composte si ha che:

$$\frac{d}{d\,\lambda} g = \sum_{i=1}^{k} \left[\frac{d}{d\,\lambda} x_i \right]^T \nabla_{x_i} f$$

In particolare, per $k = 1$ e $x(\lambda) = x_0 + \lambda s$, ovvero

$$g(\lambda) = f(x_0 + \lambda s),$$

si ha

$$\frac{d}{d\,\lambda} g = \left[\frac{d}{d\,\lambda} x \right]^T \nabla_x f = s^T \nabla_x f$$

Condizione necessaria del primo ordine

Osservazione 48 *Se x^* è un minimo locale, allora*

$$\nabla f(x^*) = 0$$

*(*condizione di stazionarietà*).*

Dimostrazione Si consideri l'espansione in forma di Taylor attorno a x^*:

$$f(x^* + \varepsilon p) = f(x^*) + \varepsilon \nabla f(x^*)^T p + o(\varepsilon).$$

Per assurdo, sia $\nabla f(x^*) \neq 0$. Allora esiste $p \in \mathbb{R}^n$ tale che $\nabla f(x^*)^T p < 0$ (esempio: $p = -\nabla f(x^*)$). Quindi, per ε sufficientemente piccolo:

$$f(x^* + \varepsilon p) < f(x^*),$$

il che contraddice l'ottimalità locale di x^*.

Si noti che *non* è una condizione sufficiente, come dimostra il seguente esempio:

$$f(x) = -x^2,$$

dove $\bar{x} = 0$ è un punto stazionario ma non è un minimo locale (è un punto di massimo).

Condizione necessaria del secondo ordine

Una condizione necessaria del secondo ordine è la seguente.

Osservazione 49 *Se x^* è un minimo locale, allora:*

- $\nabla f(x^*) = 0$*;*
- $\nabla^2 f(x^*)$ *è semidefinita positiva.*

Dimostrazione Abbiamo già dimostrato che si deve avere $\nabla f(x^*) = 0$. Quindi, dato $p \in \mathbb{R}^n$ con $\|p\| = 1$, avremo:

$$f(x^* + \varepsilon p) = f(x^*) + \frac{1}{2}\varepsilon^2 p^T \nabla^2 f(x^*) p + o(\varepsilon^2).$$

Per assurdo, sia $\nabla^2 f(x^*)$ non semidefinta positiva. Allora, $\exists\ p$ tale che

$$p^T \nabla^2 f(x^*) p < 0.$$

Quindi, per ε sufficientemente piccolo:

$$f(x^* + \varepsilon p) < f(x^*),$$

il che contraddice l'ottimalità locale di x^*.

La condizione non è sufficiente come dimostra l'esempio $f(x) = x^3$ per il quale si ha che in $\bar{x} = 0$ la condizione è soddisfatta ma il punto non è di minimo.

Condizione sufficiente del secondo ordine

Osservazione 50 *Un punto x^* che soddisfi le seguenti condizioni è un minimo locale* forte*:*

- $\nabla f(x^*) = 0$*;*
- $\nabla^2 f(x^*)$ *è definita positiva.*

Dimostrazione Per ogni matrice definita positiva A, si ha che:

$$x^T A x \geq \lambda_{\min}(A) \|x\|_2^2,$$

dove $\lambda_{\min}(A) > 0$ è il più piccolo autovalore di A. Da:

$$f(x^* + \varepsilon p) = f(x^*) + \frac{1}{2}\varepsilon^2 p^T \nabla^2 f(x^*) p + o(\varepsilon^2).$$

segue che per ogni p tale che $\|p\|_2 = 1$ e per ogni $\varepsilon > 0$ sufficientemente piccolo:

$$f(x^* + \varepsilon p) \geq f(x^*) + \frac{1}{2}\varepsilon^2 \lambda_{\min}(\nabla^2 f(x^*)) + o(\varepsilon^2) > f(x^*).$$

La condizione non è necessaria, come dimostra l'esempio $f(x) = x^4$, dove il punto $x^* = 0$ è di minimo locale forte, ma $\nabla^2 f(0)$ non è definita positiva.

16.1.2 Il caso convesso

Nel caso convesso la condizione di stazionarietà:

$$\nabla f(x^*) = 0,$$

è necessaria *e* sufficiente perché x^* sia un minimo locale. Inoltre, valgono le seguenti osservazioni.

Osservazione 51 *Se f è una funzione convessa, allora ogni minimo locale è anche globale.*

Dimostrazione La dimostrazione viene fatta per assurdo. Sia x^* un minimo locale non globale, ovvero $\exists\ \bar{x}$ tale che $f(\bar{x}) < f(x^*)$. Allora $\forall\ \lambda \in (0,1)$:

$$f(\lambda x^* + (1-\lambda)\bar{x}) \leq \lambda f(x^*) + (1-\lambda) f(\bar{x}) < f(x^*),$$

il che contraddice l'ottimalità locale di x^*.

Osservazione 52 *Se f è strettamente convessa ed esiste un minimo globale, esso è anche l'unico.*

Dimostrazione Ancora per assurdo: siano $x^*, \bar{x}$, $x^* \neq \bar{x}$, due minimi globali distinti. Si avrà che $f(\bar{x}) = f(x^*)$. Inoltre, $\forall\ \lambda \in (0,1)$:

$$f(\lambda x^* + (1-\lambda)\bar{x}) < \lambda f(x^*) + (1-\lambda) f(\bar{x}) = f(x^*),$$

il che contraddice l'ottimalità globale di x^*.

16.1.3 Il caso quadratico

Sia data una funzione quadratica:

$$f(x) = \frac{1}{2} x^T Q x + c^T x,$$

dove Q può essere sempre presa simmetrica. Si ha:

$$\nabla f(x) = Qx + c \quad \nabla^2 f(x) = Q.$$

Quindi:

- f convessa $\Leftrightarrow$ Q semidefinita positiva;
- f strettamente convessa $\Leftrightarrow$ Q definita positiva.

Nel caso quadratico strettamente convesso per individuare l'unica soluzione ottima, basta imporre la condizione di stazionarietà:

$$\nabla f(x^*) = 0 \quad \text{ovvero} \quad Qx^* + c = 0,$$

da cui:

$$x^* = -Q^{-1}c,$$

è l'unico ottimo locale e globale per questo problema.

16.2 Algoritmi di ricerca locale per il caso non vincolato

Per l'ottimizzazione non lineare in assenza di vincoli si riconoscono due grandi categorie di metodi:

- metodi linesearch;
- metodi trust region.

Entrambi i metodi sono iterativi, ovvero generano una sequenza $\{x_k\}$ di punti.

16.2.1 Metodi linesearch

Lo schema generale dei metodi linesearch è il seguente:

- Si individui una direzione di discesa d_k (ovvero, $\nabla f(x_k)^T d_k < 0$);
- Si individui, tramite una ricerca lungo la semiretta con origine x_k e direzione d_k, uno scalare $\alpha_k > 0$ tale che

$$f(x_k + \alpha_k d_k) < f(x_k);$$

- si ponga $x_{k+1} = x_k + \alpha_k d_k$ e $k = k + 1$.

Diverse scelte di d_k conducono ad algoritmi con proprietà di convergenza diverse.

Antigradiente Nel metodo dell'antigradiente si sceglie come direzione di discesa quella che garantisce la rapidità di discesa massima, ovvero:

$$d_k = -\nabla f(x_k).$$

Newton Nel metodo di Newton si prende come direzione di discesa il minimo (se esiste) della funzione quadratica ottenuta dall'espansione in forma di Taylor di f attorno a x_k troncata al secondo odrine, ovvero

$$d_k \in \arg\min_{d \in \mathbb{R}^n} [f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d]$$

o, equivalentemente:

$$d_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k).$$

NOTA BENE: non è detto che d_k sia definita e, qualora lo sia, che sia una direzione di discesa, ma è certamente una direzione di discesa nelle vicinanze di un minimo locale che soddisfa le condizioni sufficienti del secondo ordine.

Quasi-Newton Si utilizza un'approssimazione B_k dell'Hessiano $\nabla^2 f(x_k)$:

$$d_k = -[B_k]^{-1} \nabla f(x_k).$$

A ogni iterazione, l'approssimazione viene aggiornata in modo che soddisfi alcune proprietà. Viene intanto richiesto che soddisfi una condizione detta *condizione quasi-newtoniana*. Sia $s_k = x_{k+1} - x_k$ e

$$\nabla f(x_{k+1}) \approx \nabla f(x_k) + \nabla^2 f(x_k) s_k,$$

l'espansione in forma di Taylor, troncata al primo ordine, del *gradiente* della f attorno a x_k. Indicando con

$$\Delta_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

si richiederà quindi che:

$$B_{k+1} s_k = \Delta_k$$

Inoltre viene richiesto che:

- B_{k+1} sia simmetrica (come lo è l'Hessiano);
- B_{k+1} sia definita positiva (si garantisce che la direzione di ricerca sia di discesa).

Tipicamente si cercano aggiornamenti il più piccolo possibile': $B_{k+1} - B_k$ è una matrice a rango molto basso (uno o due).

Nei metodi quasi-Newton una scelta usuale è porre $B_0 = I$ (matrice identica), ovvero il primo passo è lungo la direzione dell'antigradiente. Un possibile aggiornamento, che dà origine al metodo BFGS, è il seguente:

$$B_{k+1} = B_k + \frac{\nabla f(x_k) \nabla f(x_k)^T}{\nabla f(x_k)^T d_k} - \frac{\Delta_k \Delta_k^T}{\alpha_k \Delta_k^T d_k}$$

Esiste anche una versione limited-memory BFGS, utile per problemi su larga scala, dove la matrice B_k *non* può essere memorizzata per problemi di occupazione di memoria, ma vengono memorizzati solo vettori che identificano un certo numero limitato di aggiornamenti quasi-newtoniani rispetto alla matrice identica.

16.2.2 Convergenza globale degli algoritmi

Definizione 17 *Si dice che un algoritmo, che genera la sequenza di punti* $\{x_k\}$*, ha* convergenza globale *se, dato un punto iniziale* x_0*, tutti i punti di accumulazione della sequenza* $\{x_k\}$ *sono* punti stazionari*, ovvero*

$$\|\nabla f(x_k)\| \to 0.$$

Si noti come ci si accontenti di dimostrare la convergenza a un punto stazionario e non a un minimo locale. Non si dimentichi che lo stabilire se un punto è un minimo locale risulta essere un problema difficile. Tuttavia, nella pratica, il fatto che gli algoritmi generino sequenze di punti con valori della funzione decrescenti, tipicamente garantisce che i punti stazionari a cui si converge sono anche minimi locali.

Si può dimostrare che la convergenza globale si ha sotto le seguenti condizioni:

- l'insieme di livello:

  $$L(f(x_0)) = \{x \in \mathbb{R}^n \ : \ f(x) \leq f(x_0)\}$$

  è chiuso e limitato;

- a ogni iterazione si ha una sufficiente decrescita della funzione, cioè $f(x_k) - f(x_{k+1})$ è sufficientemente grande;

- il passo compiuto $x_{k+1} - x_k$ è sufficientemente distante dall'ortogonalità rispetto al gradiente $\nabla f(x_k)$.

Tra le condizioni di sufficiente decrescita abbiamo quelle dette di Armijo-Goldstein. Indicata con d la direzione di ricerca scelta, sia $\bar{\alpha}$ il più piccolo valore per cui $f(x_k + \bar{\alpha}d) = f(x_k)$. Per garantire una sufficiente decrescita, si vuole evitare un passo troppo vicino a 0 o troppo vicino a $\bar{\alpha}$. Si fissi un $\rho \in (0, 1/2)$. Per impedire che il passo non sia troppo vicino a 0 imponiamo:

$$f(x_k + \alpha d) \geq f(x_k) + \alpha(1 - \rho)\nabla f(x_k)^T d.$$

Per impedire che sia non troppo vicino a $\bar{\alpha}$ imponiamo:

$$f(x_k + \alpha d) \leq f(x_k) + \alpha\rho\nabla f(x_k)^T d.$$

Quindi, il passo α_k soddisfa:

$$-\rho\alpha_k\nabla f(x_k)^T d \leq f(x_k) - f(x_k + \alpha_k d) \leq -(1 - \rho)\alpha_k\nabla f(x_k)^T d$$

Per quanto riguarda la non ortogonalità rispetto al gradiente, sia θ_k l'angolo tra il gradiente in x_k e il passo $s_k = x_{k+1} - x_k$. Si ha:

$$\cos(\theta_k) = \frac{\nabla f(x_k)^T s_k}{\|\nabla f(x_k)\|\|s_k\|}.$$

Per impedire l'ortogonalità rispetto al gradiente si impone:

$$\sum_{k=1}^{\infty} \cos^2(\theta_k) = +\infty,$$

cioè $\{\theta_k\}$ può anche convergere a $\pi/2$ ma deve farlo in modo sufficientemente lento.

16.2.3 Convergenza locale degli algoritmi

Definizione 18 *Dato un minimo locale $\overline{x}$ e un punto x_0 sufficientemente vicino a $\overline{x}$, la velocità di convergenza locale può essere:*

- *lineare con coefficiente $r < 1$:*

$$\|\mathbf{x}_{k+1} - \overline{\mathbf{x}}\|/\|\mathbf{x}_k - \overline{\mathbf{x}}\| \to r$$

- *superlineare:*

$$\|\mathbf{x}_{k+1} - \overline{\mathbf{x}}\|/\|\mathbf{x}_k - \overline{\mathbf{x}}\| \to 0$$

- *quadratica:*

$$\|\mathbf{x}_{k+1} - \overline{\mathbf{x}}\|/\|\mathbf{x}_k - \overline{\mathbf{x}}\|^2 \to A > 0.$$

Il metodo dell'antigradiente ha velocità di convergenza locale lineare. Se applicato a una funzione quadratica convessa $f(x) = c^T x + (1/2)x^T Hx$, con H simmetrica definita positiva e minimo in x^*, si dimostra che

$$\mid f(x_{k+1}) - f(x^*) \mid \leq \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^2 \mid f(x_k) - f(x^*) \mid$$

dove $\lambda_{\max}$ e $\lambda_{\min}$ sono, rispettivamente, il massimo e il minimo autovalore di H. Si noti che se $\lambda_{\max}/\lambda_{\min}$ è molto grande, la convergenza è lineare ma molto lenta.

Esempio 51 *Si consideri la funzione*

$$f(x, y) = Mx^2 + y^2,$$

con $M > 0$. Si può facilmente dimostrare che il punto di ottimo locale e globale di questa funzione è l'origine $(0, 0)$.Prendiamo ora il punto

$$\mathbf{x}_0 = \alpha(1/M, 1).$$

Il gradiente è $\nabla f(x, y) = (2Mx, 2y)$ e quindi nel punto $\mathbf{x}_0$ la direzione di ricerca è l'antigradiente $-\alpha(2, 2)$ e il nuovo punto viene cercato lungo la semiretta:

$$\alpha(1/M, 1) - \lambda\alpha(2, 2) \quad \lambda \geq 0.$$

Possiamo facilmente trovare il minimo della funzione:

$$q(\lambda) = f(\alpha(1/M - 2\lambda), \alpha(1 - 2\lambda)) = M\alpha^2(1/M - 2\lambda)^2 + \alpha^2(1 - 2\lambda)^2 \quad \lambda \geq 0.$$

Il minimo è il valore $\lambda_0 = \frac{1}{M+1}$*, indipendente da* α *e il nuovo punto* $\mathbf{x}_1$ *è il seguente:*

$$\mathbf{x}_1 = \alpha \frac{M-1}{M+1}(-1/M, 1).$$

Con il nuovo punto si può ripetere quanto visto sopra e si ottiene:

$$\mathbf{x}_2 = \alpha \left(\frac{M-1}{M+1}\right)^2 (1/M, 1),$$

Notando che

$$\mathbf{x}_2 = \left(\frac{M-1}{M+1}\right)^2 \mathbf{x}_0,$$

avremo alla $2k$*-esima iterazione:*

$$\mathbf{x}_{2k} = \left(\frac{M-1}{M+1}\right)^{2k} \mathbf{x}_0, \quad \forall\ k \in \mathbb{N}$$

Questo ci dice che a ogni iterazione la distanza dall'ottimo (l'origine $(0, 0)$*) viene ridotta del fattore*

$$\frac{M-1}{M+1}$$

che al crescere di M *tende a 1, confermando quindi che il metodo dell'antigradiente ha convergenza locale lineare con un coefficiente* r *che può essere reso arbitrariamente vicino a 1.*

Sotto opportune ipotesi (in particolare si richiede che nel minimo locale $\overline{x}$ l'Hessiano sia definito positivo), il metodo di Newton ha velocità di convergenza locale quadratica. D'altro canto non si ha alcuna garanzia di convergenza globale (in certe iterazioni la direzione di Newtono può non essere definita o, anche se definita, non essere una direzione di discesa).

Per i metodi quasi-newtoniani si riesce di solito a dimostrare la convergenza superlineare. Inoltre, per il metodo BFGS si riesce a dimostrare:

- la convergenza in al più n iterazioni su funzioni quadratiche, *se* si fanno ricerche lineari esatte;
- la convergenza globale per funzioni convesse, ma *non* per funzioni più generali.

Un interessante risultato, noto come caratterizzazione di Dennis-Morè, è il seguente.

Teorema 10 *Data una sequenza $\{x_k^{\mathcal{A}}\}$ generata da un metodo $\mathcal{A}$, indichiamo con*

$$s_k^{\mathcal{A}} = x_{k+1}^{\mathcal{A}} - x_k^{\mathcal{A}},$$

il passo all'iterazione k e con $s_k^{\mathcal{N}}$ il passo che verrebbe compiuto dal metodo di Newton. Si dimostra che $\mathcal{A}$ ha convergenza superlineare se e solo se

$$s_k^{\mathcal{A}} = s_k^{\mathcal{N}} + o(\|s_k^{\mathcal{N}}\|),$$

ovvero il passo è asintoticamente pari a quello di Newton.

16.3 Metodi trust-region

Si tratta di metodi iterativi in cui l'iterato successivo x_{k+1} viene determinato risolvendo un opportuno sottoproblema basato sull'iterato corrente x_k. Nel sottoproblema si minimizza un modello quadratico $M_k(x)$ che approssima f in una sfera centrata in x_k e con un certo raggio ρ_k:

$$\tilde{x}_k \in \arg \min_{x \,:\, \|x - x_k\|_2 \leq \rho_k} M_k(x).$$

Quindi si pone

$$x_{k+1} = \begin{cases} x_k & \text{se } f(\tilde{x}_k) \geq f(x_k) \\ \tilde{x}_k & \text{altrimenti} \end{cases}$$

Una tipica scelta per il modello quadratico è:

$$M_k(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T S_k (x - x_k),$$

dove S_k coincide con $\nabla^2 f(x_k)$ o è una sua approssimazione (come nei metodi quasi-newtoniani). A ogni iterazione il raggio ρ_k viene aggiornato in un nuovo valore ρ_{k+1} secondo le regole seguenti. Sia

$$r_k = \frac{f(\tilde{x}_k) - f(x_k)}{M_k(\tilde{x}_k) - M_k(x_k)},$$

il rapporto tra la riduzione effettiva e quella sul modello. Dati $0 < \delta_1 < \delta_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$ si pone:

$$\rho_{k+1} = \begin{cases} \gamma_1 \|\tilde{x}_k - x_k\| & \text{se } r_k < \delta_1 \\ \gamma_2 \rho_k & \text{se } r_k > \delta_2 \text{ e } \|\tilde{x}_k - x_k\| = \rho_k \\ \rho_k & \text{altrimenti} \end{cases}$$

In pratica, se il modello è affidabile, tendiamo a estendere la regione in cui applicarlo, se non lo è la riduciamo. Valgono i seguenti risultati di convergenza.

Osservazione 53 *Sia $\{x_k\}$ la sequenza generata da un metodo trust region con $S_k = \nabla^2 f(x_k)$. Se $\{x_k\} \subseteq B$, dove B è un insieme limitato (ad esempio, ciò accade se l'insieme di livello $L(f(x_0))$ è limitato), allora esiste un punto di*

accumulazione della sequenza che soddisfa le condizioni necessarie *del secondo ordine.*

Se il punto di accumulazione soddisfa le condizioni sufficienti *del secondo ordine, allora la velocità di convergenza locale è quadratica,* $r_k \to 1$ *e il vincolo* $\|x - x_k\| \leq \rho_k$ *diventa inattivo (ovvero* $\|\tilde{x}_k - x_k\| < \rho_k$*) per* k *sufficientemente grande.*

16.4 Il caso vincolato: condizioni di ottimalità

Diamo innnanzitutto la definizione di direzione ammissibile.

Definizione 13 *Sia* x' *un punto ammissibile e* $\{x_k\}$ *una sequenza di punti ammissibili convergente a* x' *con*

$$x_k - x' = \delta_k s_k \quad \|s_k\| = 1,\ \delta_k \to 0.$$

Sia $s_k \to s$*. La direzione* s *viene detta* direzione ammissibile *rispetto a* x'*. L'insieme delle direzioni ammissibili rispetto a* x' *viene indicato con*

$$\mathcal{F}(x')$$

Sia

$$\mathcal{A}(x') = \{i \in K_2 \ :\ c_i(x') = 0\},$$

l'insieme dei *vincoli attivi* in x' (vincoli di diseguaglianza soddisfatti come uguaglianza in x').

Se linearizziamo i vincoli con un'espansione di Taylor troncata al primo ordine attorno a x', questi avranno la seguente forma:

$$\begin{array}{ll} \nabla c_i(x')^T(x - x') = 0 & i \in K_1 \\ c_i(x') + \nabla c_i(x')^T(x - x') \geq 0 & i \in K_2 \end{array}$$

L'insieme $F(x')$ delle direzioni ammissibili dei vincoli linearizzati è dato da tutte le direzioni s tali che

$$\begin{array}{ll} \nabla c_i(x')^T s = 0 & i \in K_1 \\ \nabla c_i(x')^T s \geq 0 & i \in K_2 \cap \mathcal{A}(x') \end{array}$$

In generale si ha

$$F(x') \supseteq \mathcal{F}(x').$$

con la possibilità che non valga l'uguaglianza.

Esempio 52 *Si considerino i seguenti due vincoli di uguaglianza:*

$$\begin{array}{rcl} c_1(x_1, x_2) & = & (x_1 - 1)^2 + x_2^2 - 1 \\ c_2(x_1, x_2) & = & (x_1 + 1)^2 + x_2^2 - 1 \end{array}$$

Per $x' = (0, 0)$ *ogni vettore* $(0, \delta)$ *è in* $F(x')$ *ma essendo* x' *l'unico punto ammissibile, si ha che* $\mathcal{F}(x')$ *è vuoto.*

Definizione 14 *Chiamiamo* constraint qualifications *in x' delle condizioni per cui*

$$F(x') = \mathcal{F}(x').$$

Tra queste abbiamo:

- *tutti i vincoli c_i sono lineari;*
- *i gradienti $\nabla c_i(x')$ dei vincoli in $K_1 \cup \mathcal{A}(x')$ sono tra loro linearmente indipendenti.*

Esempio 53 *Nell'esempio i vincoli sono ovviamente non lineari. Inoltre, si ha che:*

$$\nabla c_1(0,0) = (-2,0), \quad \nabla c_2(0,0) = (2,0)$$

non sono linearmente indipendenti.

16.4.1 Condizione necessaria del primo ordine

Vediamo ora di definire una condizione necessaria perché un punto sia un minimo locale. Indichiamo l'insieme delle direzioni di discesa in x' in questo modo:

$$\mathcal{D}(x') = \{s \ : \ \nabla f(x')^T s < 0\}.$$

Se x^* è un minimo locale, allora

$$\mathcal{F}(x^*) \cap \mathcal{D}(x^*) = \emptyset.$$

Se in x^* valgono le constraint qualifications, allora condizione necessaria perché x^* sia un minimo locale è che:

$$F(x^*) \cap \mathcal{D}(x^*) = \emptyset.$$

Ma vediamo ora di esprimere tale condizione in un'altra forma. Per questo abbiamo bisogno di dimostrare il *lemma di Farkas*.

Lemma 2 *Dati i vettori $a_1, \ldots, a_m$ e g si ha che*

$$U = \{s \ : \ s^T g < 0, \ s^T a_i \geq 0, \ i = 1, \ldots, m\}$$

è vuoto se e solo se $\exists$ *moltiplicatori $\lambda_i \geq 0$ tali che*

$$g = \sum_{i=1}^{m} \lambda_i a_i.$$

Dimostrazione *Se*: facile $\rightarrow$

$$s^T g = \sum_{i=1}^{m} \underbrace{\lambda_i}_{\geq 0} \underbrace{s^T a_i}_{\geq 0} \geq 0$$

Solo se: sia

$$C = \{v \ : \ v = \sum_{i=1}^{m} \lambda_i a_i, \ \lambda_i \geq 0, i = 1, \ldots, m\}$$

un cono poliedrale e si ipotizzi che $g \notin C$. Allora esiste un iperpiano $s^T x = 0$ (con normale s) che separa g da C, ovvero:

$$s^T a_i \geq 0, \quad s^T g < 0,$$

come richiesto.

Nella dimostrazione vista sopra abbiamo postulato l'esistenza di un un iperpiano $s^T x = 0$ detto *iperpiano separatore*. La dimostriamo ora nella seguente osservazione.

Osservazione 54 *sia*

$$C = \{v \ : \ v = \sum_{i=1}^{m} \lambda_i a_i, \ \lambda_i \geq 0, i = 1, \ldots, m\}$$

un cono poliedrale e si ipotizzi che $g \notin C$. Allora esiste un iperpiano $s^T x = 0$ (con normale s) che separa g da C, ovvero:

$$s^T a_i \geq 0, \quad s^T g < 0,$$

Dimostrazione Si consideri $\min_{x \in C} \|g - x\|_2^2$. Dato $x_1 \in C$ si può imporre, senza perdere soluzioni ottime, il vincolo $\|g - x\|_2 \leq \|g - x_1\|_2$ che rende la regione ammissibile limitata. Essendo l'obiettivo una funzione continua, il teorema di Weierstrass garantisce l'esistenza di un punto di minimo $\hat{g}$. Dal momento che $\lambda \hat{g} \in C \ \forall \ \lambda \geq 0$, si ha che $\|\lambda \hat{g} - g\|_2^2$ ha un minimo in $\lambda = 1$. Quindi,

$$\frac{d}{d \, \lambda} \|\lambda \hat{g} - g\|_2^2 \, |_{\lambda = 1} = \hat{g}^T (\hat{g} - g) = 0.$$

Sia $x \in C$; allora, per convessità, $\forall \ \theta \in (0, 1)$:

$$\hat{g} + \theta (x - \hat{g}) \in C$$

e quindi:

$$\|\theta (x - \hat{g}) + \hat{g} - g\|_2^2 \geq \|\hat{g} - g\|_2^2.$$

e:

$$\theta^2 \|x - \hat{g}\|_2^2 + \theta (x - \hat{g})^T (\hat{g} - g) \geq 0$$

Prendendo il limite per $\theta \to 0$, si ha

$$(x - \hat{g})^T (\hat{g} - g) = x^T (\hat{g} - g) \geq 0.$$

Si prenda ora $s = \hat{g} - g \neq 0$ poiché $g \notin C$. Si ha

$$s^T x \geq 0 \quad \forall \ x \in C, \quad s^T g = -s^T s < 0,$$

come si voleva dimostrare.

Vediamo ora una conseguenza del lemma di Farkas. Osserviamo che:

$$\mathcal{D}(x^*) \cap F(x^*) = \left\{ \begin{array}{lll} s \ : & s^T \nabla f(x^*) < 0 & \\ & s^T \nabla c_i(x^*) = 0 & i \in K_1 \\ & s^T \nabla c_i(x^*) \geq 0 & i \in \mathcal{A}(x^*) \end{array} \right\}$$

è vuoto *se e solo se*:

$$\left\{ \begin{array}{lll} s \ : & s^T \nabla f(x^*) < 0 & \\ & s^T \nabla c_i(x^*) \geq 0, \ -s^T \nabla c_i(x^*) \geq 0 & i \in K_1 \\ & s^T \nabla c_i(x^*) \geq 0 & i \in \mathcal{A}(x^*) \end{array} \right\}$$

il che è vero se e solo se (lemma di Farkas) $\exists \ \lambda_i^+, \lambda_i^- \geq 0$, $i \in K_1$, $\mu_i^* \geq 0$, $i \in K_2$ (con $\mu_i^* = 0$ se $i \notin \mathcal{A}(x^*)$), tali che:

$$\nabla f(x^*) = \sum_{i \in K_1} (\lambda_i^+ - \lambda_i^-) \nabla c_i(x^*) + \sum_{i \in K_2} \mu_i^* \nabla c_i(x^*),$$

o, equivalentemente, esistono λ_i^*, $i \in K_1$ e $\mu_i^* \geq 0$, $i \in K_2$ tali che

$$\nabla f(x^*) = \sum_{i \in K_1} \lambda_i^* \nabla c_i(x^*) + \sum_{i \in K_2} \mu_i^* \nabla c_i(x^*).$$

($(\lambda_i^+ - \lambda_i^-)$ con $\lambda_i^+, \lambda_i^- \geq 0$ vengono sostituiti con λ_i non vincolato in segno). Tutto questo conduce alle em condizioni di Karush-Kuhn-Tucker (KKT) (condizioni necessarie del primo ordine di ottimalità locale).

Teorema 11 *Indicata con*

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{i \in K_1} \lambda_i c_i(x) - \sum_{i \in K_2} \mu_i c_i(x),$$

la funzione Lagrangiana*, se x^* è un minimo locale, allora esistono* moltiplicatori di Lagrange λ^*, μ^**, tali che*

- $\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$*;*
- $c_i(x^*) = 0$*,* $i \in K_1$*;*
- $c_i(x^*) \geq 0$*,* $i \in K_2$*;*
- $\mu^* \geq 0$*;*
- $\mu_i^* c_i(x^*) = 0$*,* $\forall \ i \in K_2$ *(condizioni di complementarità).*

Vale la pena dare un'interpretazione del significato dei moltiplicatori di Lagrange. Lo faremo solo per il caso di vincoli di uguaglianza (per i vincoli di disuguaglianza si ha un'interpretazione del tutto simile). Si consideri la perturbazione

$$c_i(x) = \varepsilon_i$$

dell'i-esimo vincolo di uguaglianza. Consideriamo ora la funzione Lagrangiana

$$\mathcal{L}(x, \lambda, \varepsilon) = f(x) - \sum_{i \in K_1} \lambda_i (c_i(x) - \varepsilon_i).$$

Siano $x(\varepsilon), \lambda(\varepsilon)$ la soluzione e il moltiplicatore di Lagrange del problema perturbato. Si ha $f(x(\varepsilon)) = \mathcal{L}(x(\varepsilon), \lambda(\varepsilon), \varepsilon)$ (valore ottimo del problema perturbato) e

$$\frac{d\ f}{d\ \varepsilon_i} = \frac{d\ \mathcal{L}}{\varepsilon_i} = \frac{\partial\ x^T}{\partial\ \varepsilon_i} \underbrace{\nabla_x \mathcal{L}}_{=0} + \frac{\partial\ \lambda^T}{\partial\ \varepsilon_i} \underbrace{\nabla_\lambda \mathcal{L}}_{=0} + \frac{\partial\ \mathcal{L}}{\partial\ \varepsilon_i} = \lambda_i$$

Quindi, i moltiplicatori di Lagrange misurano la rapidità di variazione del valore ottimo in corrispondenza di perturbazioni dei vincoli.

Come nel caso non vincolato, anche in quello vincolato i problemi convessi hanno proprietà particolari. Ricordiamo intanto che chiamiamo convessi quei problemi per cui:

- f convessa;
- $K_1 = \emptyset$;
- c_i, $i \in K_2$ sono funzioni concave (equivalentemente: $-c_i$ sono funzioni convesse).

I problemi di programmazione convessa sono caratterizzati dal fatto che:

- ogni minimo locale è anche globale;
- se f è strettamente convessa, un minimo globale (se esiste) è unico;
- le condizioni KKT sono *necessarie e sufficienti* per identificare i minimi locali (globali).

16.4.2 Condizioni del secondo ordine

Sotto l'ipotesi del primo ordine

$$\nabla f(x^*) = \sum_{i \in K_1} \lambda_i^* \nabla c_i(x^*) + \sum_{i \in K_2} \mu_i^* \nabla c_i(x^*),$$

se $\mu_j^* > 0$, allora le direzioni ammissibili s per cui $\nabla c_j(x^*)^T s > 0$, sono di crescita:

$$\nabla f(x^*)^T s = \sum_{i \in K_1} \lambda_i^* \nabla c_i(x^*)^T s + \sum_{i \in K_2} \mu_i^* \nabla c_i(x^*)^T s \geq \mu_j^* \nabla c_j(x^*)^T s > 0.$$

Quindi, sotto l'ipotesi che valgano delle constraint qualifications in x^*, possiamo restringerci alle direzioni

$$G(x^*) = \{s \ :\ \nabla c_i(x^*)^T s = 0,\ i \in K_1 \text{ o } \mu_i^* > 0,\ \nabla c_i(x^*)^T s \geq 0\ i \in \mathcal{A}(x^*) \text{ e } \mu_i^* = 0\}.$$

Se $x_k = x^* + \delta_k s_k$ con $\delta_k \to 0$, $s_k \to s \in G(x^*)$ e

$$c_i(x_k) = 0 \ \forall \ i \ : \ i \in K_1 \text{ o } \mu_i^* > 0,$$

si ha

$$f(x^* + \delta_k s_k) = \mathcal{L}(x^* + \delta_k s_k, \lambda^*, \mu^*)$$

Si ha anche:

$$\mathcal{L}(x^* + \delta_k s_k, \lambda^*, \mu^*) =$$
$$\underbrace{\mathcal{L}(x^*, \lambda^*, \mu^*)}_{=f(x^*)} + \delta_k \underbrace{\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*)^T s_k}_{=0} + \tfrac{1}{2}\delta_k^2 s_k^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*) s_k + o(\delta_k^2)$$

Se x^* è un minimo locale, allora $f(x_k) \geq f(x^*)$, per k sufficientemente grande. Quindi, per $k \to \infty$:

$$s^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*) s \geq 0 \quad \forall \ s \in G(x^*),$$

ovvero, se indichiamo con $Z(x^*)$ una matrice le cui colonne formano una base dell'insieme di direzioni $G(x^*)$, si ha la condizione *necessaria* del secondo ordine:

$$Z(x^*)^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*) Z(x^*) \quad \text{semidefinita positiva.}$$

Se la semidefinita positività viene sostituita con la definita positività, allora otteniamo una condizione *sufficiente* del secondo ordine. Più precisamente, se x^*, λ^*, μ^* soddisfano le condizioni KKT e

$$Z(x^*)^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*) Z(x^*) \quad \text{definita positiva,}$$

allora x^* è un minimo locale del problema.

16.4.3 Duale di Wolfe

Si consideri il seguente problema dove appare la funzione Lagrangiana:

$$\begin{array}{ll} \max_{x,\lambda,\mu} & \mathcal{L}(x, \lambda, \mu) \\ & \nabla_x \mathcal{L}(x, \lambda, \mu) = 0 \\ & \mu \geq 0 \end{array}$$

Questo viene chiamato *duale di Wolfe*. Vale il seguente teorema.

Teorema 12 *Se abbiamo un problema vincolato tale che:*

- *è di programmazione convessa;*
- *x^* è una sua soluzione con i corrispondenti moltiplicatori di Lagrange λ^*, μ^*;*

- *in x^* valgono delle constraint qualifications;*

allora (x^*, λ^*, μ^*) risolve il duale di Wolfe del problema e i valori ottimi dei due problemi coincidono.

Esempio 54 *Si consideri il problema lineare*

$$\begin{array}{ll} \min & c^T x \\ & A^T x \leq b \\ & x \geq 0 \end{array} \tag{16.2}$$

Il corrispondente duale di Wolfe è:

$$\begin{array}{ll} \max_{x,\mu,\lambda} & c^T x - \mu^T (A^T x - b) - \gamma^T x \\ & c - A\mu - \gamma = 0 \\ & \mu, \gamma \geq 0 \end{array}$$

Equivalentemente, osservando che $c = A\mu + \gamma$, si può riscrivere il duale di Wolfe in questo modo:

$$\begin{array}{ll} \max_{\mu} & \mu^T b \\ & A\mu \leq c \\ & \mu \geq 0 \end{array}$$

Si noti che questo coincide esattamente con il problema duale definito per i problemi di PL di forma (16.2) (si faccia riferimento alla Sezione 5.2) e la stessa cosa varrebbe per problemi di PL di qualsiasi altra forma. Si noti anche come i moltiplicatori di Lagrange coincidano con le variabili del problema duale. Dalla soluzione ottima μ^* del duale, si ricava immediatamente la soluzione ottima del primale a partire dalle condizioni di complementarità:

$$\gamma^{*T} x^* = (A\mu^* - c)^T x^* = 0 \qquad \mu^{*T} (A^T x^* - b) = 0$$

Esempio 55 *Si consideri il problema quadratico:*

$$\begin{array}{ll} \min_{x} & \frac{1}{2} x^T Q x + c^T x \\ & A^T x \geq b \end{array}$$

con Q simmetrica definita positiva. Il corrispondente duale di Wolfe è:

$$\begin{array}{ll} \max_{x,\mu} & \frac{1}{2} x^T Q x + c^T x - \mu^T (A^T x - b) \\ & Qx + c - A\mu = 0 \\ & \mu \geq 0 \end{array}$$

Osservando che $x = Q^{-1}(A\mu - c)$, si può riscrivere il duale di Wolfe in questo modo:

$$\begin{array}{ll} \max_{\mu} & -\frac{1}{2} \mu^T (A^T Q^{-1} A) \mu + \mu^T (b + A^T Q^{-1} c) - \frac{1}{2} c^T Q^{-1} c \\ & \mu \geq 0 \end{array}$$

Dalla soluzione ottima μ^* del duale, si ricava immediatamente la soluzione ottima del primale

$$x^* = Q^{-1}(A\mu^* - c).$$

Appendice A

Grafi

Un grafo $G = (V, A)$ è definito da una coppia di insiemi, l'insieme V dei *nodi* del grafo e l'insieme $A \subseteq V \times V$ (sottinsieme dell'insieme di tutte le possibili coppie di nodi) degli *archi* del grafo. I grafi sono oggetti matematici attraverso cui è possibile dare una rappresentazione astratta di relazioni tra entità: le entità vengono rappresentate con i nodi, mentre un arco congiunge due nodi se le entità corrispondenti a tali nodi sono in relazione tra loro. Se la relazione è simmetrica (cioè i è in relazione con j se e solo se j è in relazione con i), gli archi utilizzati saranno privi di verso (*grafo non orientato*), altrimenti all'arco dovrà essere assegnato un verso (*grafo orientato*). Per esempio, se pensiamo alle relazioni parentali, la relazione 'è fratello di' è chiaramente simmetrica, mentre non lo è la relazione 'è padre di'.

Esempio 56 *Consideriamo un grafo G con insieme di nodi*

$$V = \{a, b, c, d, e\},$$

mentre l'insieme di archi è il seguente sottinsieme di coppie di nodi in V

$$A = \{(a, b);\ (a, c);\ (b, c);\ (b, e);\ (c, d);\ (d, b)\}$$

Se tali coppie sono ordinate (e quindi, ad esempio, la coppia (a, b) è diversa dalla coppia (b, a)) il grafo è orientato, altrimenti (e quindi, ad esempio, la coppia (a, b) e la coppia (b, a) sono equivalenti tra loro) il grafo è non orientato.

Diamo ora una definizione.

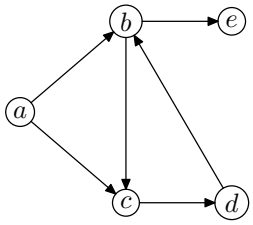
Definizione 15 *Dato un grafo orientato $G = (V, A)$ e un arco $(i, j) \in A$ diremo che il nodo i è* predecessore *del nodo j e che il nodo j è* successore *del nodo i. Nel caso di un grafo non orientato $G = (V, A)$, dato un arco $(i, j) \in A$ diremo che il nodo i e il nodo j sono tra loro* adiacenti.

Nel grafo dell'esempio, supposto orientato, il nodo a è predecessore del nodo b, mentre b è successore di a. Se il grafo fosse non orientato diremmo semplicemente che i nodi a e b sono adiacenti. Abbiamo definito un grafo G attraverso la coppia di insiemi V e A. È possibile però rappresentare un grafo anche in altri modi.

Tabella A.1:

	(a,b)	(a,c)	(b,c)	(b,e)	(c,d)	(d,b)
a	1	1	0	0	0	0
b	-1	0	1	1	0	-1
c	0	-1	-1	0	1	0
d	0	0	0	0	-1	1
e	0	0	0	-1	0	0

Rappresentazione grafica Ad ogni nodo corrisponde un punto sul piano e ogni arco (i,j) corrisponde a una linea che congiunge il punto che rappresenta il nodo i con il punto che rappresenta il nodo j. Se il grafo è orientato sulla linea si aggiunge anche una freccia da i verso j (per grafi non orientati la freccia si omette). Nella figura sottostante è mostrata la rappresentazione grafica del grafo dell'esempio, supposto orientato (nel caso non orientato si devono solo omettere le frecce).



Liste di adiacenza Ad ogni nodo si affianca una lista (eventualmente vuota) contenente tutti i suoi successori nel caso di grafo orientato, o tutti i nodi adiacenti nel caso di grafo non orientato. Per il nostro esempio, supposto orientato, le liste di adiacenza sono le seguenti:

$$a:\ (b,c)\quad b:\ (c,e)\quad c:\ (d)\quad d:\ (b)\quad e:\ \emptyset$$

mentre se lo si suppone non orientato le liste di adiacenza sono le seguenti:

$$a:\ (b,c)\quad b:\ (a,c,d,e)\quad c:\ (a,b,d)\quad d:\ (b,c)\quad e:\ (b)$$

Matrice incidenza nodo-arco Si costruisce una matrice con una riga per ogni nodo e una colonna per ogni arco. Per grafi orientati nella colonna relativa all'arco (i,j) si mette $+1$ in corrispondenza della riga i, -1 in corrispondenza della riga j e 0 in corrispondenza di tutte le altre righe. Per il nostro esempio, supposto orientato, la matrice di incidenza nodo-arco è data in Tabella A.1 Per grafi non orientati nella colonna relativa all'arco (i,j) si mette $+1$ in corrispondenza della riga i e della riga j e 0 in corrispondenza di tutte le altre righe. Per il nostro esempio, supposto non orientato, la matrice di incidenza nodo-arco è data in Tabella A.2

Introduciamo ora alcune definizioni.

Tabella A.2:

	(a,b)	(a,c)	(b,c)	(b,e)	(c,d)	(d,b)
a	1	1	0	0	0	0
b	1	0	1	1	0	1
c	0	1	1	0	1	0
d	0	0	0	0	1	1
e	0	0	0	1	0	0

Definizione 16 *Due archi che hanno un nodo in comune sono detti* adiacenti.

Nell'esempio gli archi (a,b) e (a,c) sono adiacenti.

Definizione 17 *Sia dato un grafo $G=(V,A)$. Una sequenza di $m+1$ nodi*

$$s_0 \to s_1 \to s_2 \to \cdots \to s_m$$

tali che per ogni $i=1,\ldots,m$ si ha:

$$(s_{i-1},s_i) \in A \quad \textit{oppure} \quad (s_i,s_{i-1}) \in A$$

(l'alternativa è superflua nel caso non orientato) è detto cammino *nel grafo. Il numero m di archi del cammino è detto* lunghezza del cammino. *Possiamo vedere anche un cammino di lunghezza m come una sequenza di archi a due a due adiacenti. Un cammino è detto* semplice *se nessun arco è percorso più di una volta,* elementare *se nessun nodo viene toccato più di una volta.*

Nell'esempio, il cammino $a \to b \to d \to c$ è un cammino elementare di lunghezza 3; il cammino $a \to b \to c \to d \to b \to e$ è semplice ma non elementare (il nodo b è toccato più di una volta); il cammino $a \to b \to c \to d \to b \to a \to c$ è né semplice né elementare (l'arco (a,b) è attraversato due volte).

Definizione 18 *Un cammino semplice*

$$s_0 \to s_1 \to s_2 \to \cdots \to s_m$$

in cui primo e ultimo nodo coincidono (cioè $s_m = s_0$) viene detto ciclo *di lunghezza m. Se omettendo l'ultimo nodo s_m si ottiene un cammino elementare, si parla di* ciclo elementare.

Nell'esempio $a \to b \to c \to a$ è un ciclo elementare di lunghezza 3.

Definizione 19 *In un grafo orientato un cammino o un ciclo*

$$s_0 \to s_1 \to s_2 \to \cdots \to s_m$$

($s_m = s_0$ nel caso di un ciclo) in cui tutti gli archi sono percorsi secondo il loro orientamento, ovvero si ha che per ogni $i=1,\ldots,m$:

$$(s_{i-1},s_i) \in A$$

viene detto orientato*, altrimenti si dice* non orientato.

Nell'esempio il cammino $a \to b \to c$ è orientato mentre il cammino $b \to a \to c$ è non orientato. il ciclo $a \to b \to c \to a$ è non orientato, mentre il ciclo $b \to c \to d \to b$ è orientato. Tra i cicli in un grafo una particolare rilevanza hanno i circuiti hamiltoniani.

Definizione 20 *In un grafo orientato definiamo circuito hamiltoniano un ciclo elementare orientato che tocca tutti i nodi del grafo.*

Introduciamo ora una relazione tra nodi del grafo.

Definizione 21 *Dati due nodi i e j di un grafo G, se esiste un cammino da i a j allora si dice che j 'è* accessibile *da' i.*

La relazione tra i nodi è accessibile da è una relazione di equivalenza, ovvero soddisfa le tre proprietà riflessiva, simmetrica e transitiva. Come tale, induce classi di equivalenza nell'insieme dei nodi. Tali classi di equivalenza vengono dette *componenti connesse* del grafo. Ogni componente connessa è formata da nodi tutti accessibili tra loro ma non accessibili da nodi in altre componenti.

Definizione 22 *Se il grafo contiene una sola componente connessa viene detto* connesso.

La seguente procedura consente di individuare le componenti connesse di un grafo e quindi anche di stabilire se un grafo è connesso.

Inizializzazione Si ponga $W = V$ e $r = 1$.

Passo 1 Si selezioni un nodo $i \in W$ e si ponga $S = \{i\}$ e $T_r = \emptyset$.

Passo 2 Si selezioni un nodo $j \in S$. Si rimuova j da S e si aggiungano in S tutti i nodi che non siano già contenuti in T_r di cui j è predecessore o successore, cioè

$$S = (S \setminus \{j\}) \cup \{k \notin T_r : \ (k, j) \in A \text{ o } (j, k) \in A\}.$$

Si ponga $T_r = T_r \cup \{j\}$.

Passo 3 Se $S = \emptyset$, allora si vada al Passo 4. Se $T_r \cup S = W$, si ponga $T_r = T_r \cup S$ e si vada al Passo 4. Altrimenti si ritorni al Passo 2.

Passo 4 Si ponga $W = W \setminus T_r$. Se $W = \emptyset$, allora $T_1, T_2, \ldots, T_r$ sono gli insiemi di nodi delle componenti connesse del grafo. Altrimenti si ponga $r = r + 1$ e si ritorni al Passo 1.

Come esercizio si applichi la procedura per verificare che il grafo dell'esempio è connesso.

Definizione 23 *Un grafo si dice* completo *se esiste un arco tra ogni coppia di nodi distinti.*

Il nostro grafo non è completo. Ad esempio, non c'è alcun arco tra a ed e. Lo è invece il grafo non orientato $G = (V, A)$ con

$$V = \{a, b, c, d\}$$

e

$$A = \{(a, b);\ (c, a);\ (a, d);\ (b, c);\ (d, b);\ (c, d)\}$$

Definizione 24 *Dato un grafo $G = (V, A)$ e un sottinsieme $A' \subseteq A$, un grafo $G' = (V, A')$ è detto* grafo parziale *di G. Dati $V'' \subseteq V$ e*

$$A'' \subseteq A(V'') = \{(i, j) \in A :\ i \in V'',\ j \in V''\}$$

il grafo $G'' = (V'', A'')$ viene detto sottografo *di G. In particolare, se $A'' = A(V'')$ il sottografo viene detto* sottografo indotto *da V''.*

Nell'esempio $G' = (V, A')$ con

$$A' = \{(a, b);\ (b, c);\ (b, e);\ (c, d);\ (d, b)\}$$

è un grafo parziale di G, mentre $G'' = (V'', A'')$ con $V'' = \{a, b, d\}$ e

$$A'' = \{(a, b)\}$$

è un sottografo di G. Se invece si considera $G'' = (V'', A'')$ con $V'' = \{a, b, d\}$ e

$$A'' = \{(a, b);\ (d, b)\}$$

questo è il sottografo di G indotto da V''.

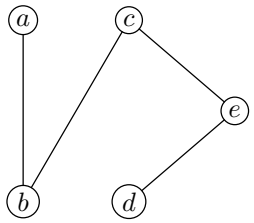
Definizione 25 *Sia dato un grafo $G = (V, A)$ con $\mid V \mid = n$ dove $\mid V \mid$ denota la cardinalità (il numero di elementi) dell'insieme V. Si dice che G è un albero se soddisfa le seguenti condizioni (equivalenti tra loro)*

1. *G è privo di cicli e connesso;*
2. *G è privo di cicli e $\mid A \mid = n - 1$;*
3. *G è connesso e $\mid A \mid = n - 1$;*
4. *esiste un unico cammino elementare che congiunge ogni coppia di nodi.*

Ad esempio, il grafo $G = (V, A)$ con

$$V = \{a, b, c, d, e\} \quad A = \{(a, b); (b, c); (c, e); (e, d)\}$$

illustrato nella figura sottostante è un albero.



Vale la seguente osservazione.

Osservazione 55 *Dato un albero, l'aggiunta di un solo arco crea esattamente un ciclo.*

Definizione 26 *Sia dato un grafo generico $G = (V, A)$. Si definisce* albero di supporto *o* spanning tree *di G un grafo parziale $T = (V, A_T)$ di G (quindi con $A_T \subseteq A$) che è un albero.*

Si noti che un albero di supporto di G deve contenere tutti i nodi di G e che in virtù del punto 2. (o del punto 3.) della Definizione 25, si dovrà avere $| A_T | = | V | -1$.

Esempio 57 *Sia dato il grafo $G = (V, A)$ con*

$$V = \{a, b, c, d\} \quad A = \{(a, b); (b, c); (b, d); (a, d); (c, d)\}$$

illustrato in Figura A.1. Un albero di supporto di G è il sottografo $T_1 = (V, A_{T_1})$ con

$$A_{T_1} = \{(b, c); (b, d); (a, d)\}$$

un altro è il sottografo $T_2 = (V, A_{T_2})$ con

$$A_{T_2} = \{(a, b); (b, c); (c, d)\}$$

I due alberi di supporto di G sono illustrati in Figura A.1 (si ignorino i numeri affiancati agli archi).

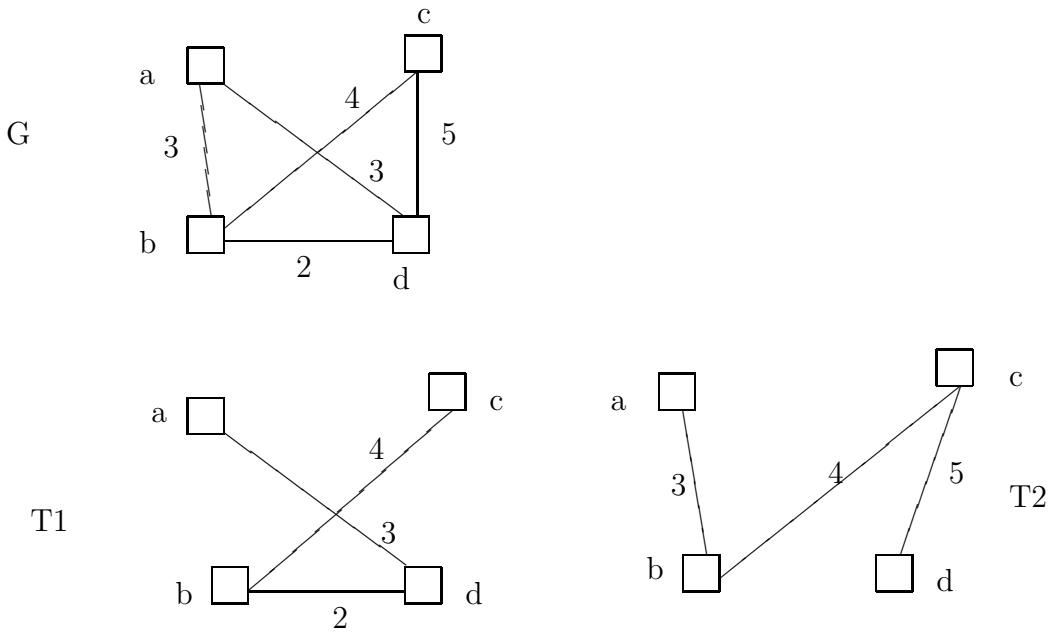


Figura A.1: Un grafo G e due suoi alberi di supporto T_1 e T_2.

Definizione 27 *Un grafo $G = (V, A)$ si dice* bipartito *se l'insieme V può essere partizionato in due sottinsieme V_1 e V_2 (quindi $V_1 \cup V_2 = V$ e $V_1 \cap V_2 = \emptyset$) tali che*

$$\forall\ (i,j) \in A: \quad i \in V_1,\ j \in V_2 \text{ oppure } i \in V_2,\ j \in V_1.$$

Un grafo bipartito si dice completo *se per ogni coppia di nodi in V_1 e V_2 esiste un arco che li congiunge.*

Vale la seguente osservazione.

Osservazione 56 *Un grafo è bipartito se e solo se non contiene cicli di lunghezza dispari.*

La seguente procedura consente di stabilire se un grafo è bipartito.

Passo 0 Si ponga $W = V$, $C_1 = C_2 = \emptyset$.

Passo 1 Si selezioni un nodo $i \in W$ e si ponga $T_1 = \{i\}$, $C_1 = C_1 \cup T_1$.

Passo 2 Si ponga

$$T_2 = \{k \in V \setminus C_2 \ :\ \exists\ i \in T_1 \text{ tale che } (i,k) \in A \text{ oppure } (k,i) \in A\}$$

Sia $C_2 = C_2 \cup T_2$.

Passo 3 Si ponga

$$T_1 = \{k \in V \setminus C_1 \ :\ \exists\ i \in T_2 \text{ tale che } (i,k) \in A \text{ oppure } (k,i) \in A\}$$

Sia $C_1 = C_1 \cup T_1$.

Passo 4 Se $C_1 \cap C_2 \neq \emptyset$, allora il grafo non è bipartito. Altrimenti si vada al Passo 5.

Passo 5 Si ponga $W = W \setminus (C_1 \cup C_2)$. Se $W = \emptyset$ e $T_1 = \emptyset$, allora il grafo è bipartito con $V_1 = C_1$ e $V_2 = C_2$. Altrimenti, se $T_1 = \emptyset$, si ritorni al Passo 1, se $T_1 \neq \emptyset$ si ritorni al Passo 2.

Nel nostro esempio selezionando inizialmente il nodo a e ponendolo in T_1 e C_1 avremo

$$\frac{C_1\ a}{C_2}$$

con $T_1 = \{a\}$. Al Passo 2 avremo

$$\frac{C_1\ a}{C_2\ b\ c}$$

con $T_2 = \{b, c\}$ e $C_2 = \{b, c\}$. Al Passo 3 avremo

$$\frac{C_1\ a\ e\ d\ c}{C_2\ b\ c}$$

con $T_1 = \{e, d, c, b\}$ e $C_1 = \{a, e, d, c, b\}$. Al Passo 4 notiamo che $C_1 \cap C_2 = \{b, c\} \neq \emptyset$ e quindi possiamo concludere che il grafo non è bipartito.

Appendice B

Breve introduzione ad AMPL

Come abbiamo visto, il primo passo per risolvere un problema reale attraverso strumenti matematici consiste nel passare dalla *descrizione a parole* del problema al *modello matematico* dello stesso. Questo passaggio verrà ora illustrato

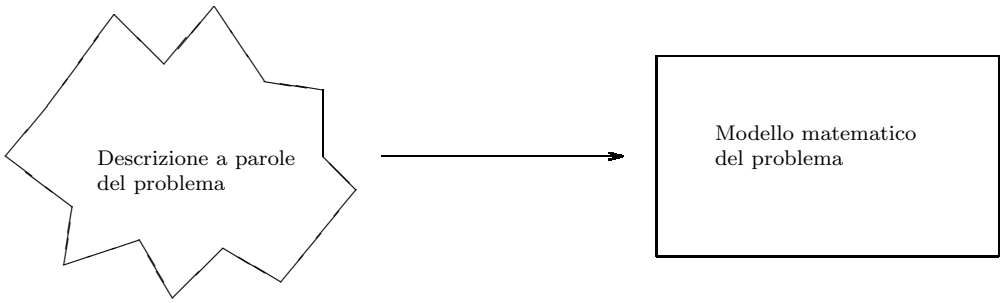


Figura B.1: Il passaggio dalla descrizione a parole al modello matematico.

in un caso particolare, il problema della dieta. La descrizione a parole di tale problema è la seguente.

Descrizione a parole Sia dato un insieme $PROD$ di prodotti ed un insieme $SOST$ di sostanze nutritive. Si sa che in un'unità del prodotto j si trova una quantità $quant\_unit_{ij}$ della sostanza nutritiva i. Inoltre si sa che il costo di un'unità di prodotto j è pari a $cost\_unit_j$. Tenendo conto che una dieta deve contenere una quantità minima $quant\_min_i$ di ciascuna sostanza nutritiva, si vuole determinare una dieta che abbia costo minimo.

Da tale descrizione si passa al modello matematico che risulterà essere un modello di PL.

Modello matematico Ad ogni prodotto j si associa una *variabile* x_j indicante la quantità di prodotto da inserire nella dieta. L' *obiettivo* è quello di minimizzare il costo complessivo della dieta, cioè

$$\min \sum_{j \in PROD} cost\_unit_j * x_j. \tag{B.1}$$

Inoltre abbiamo i vincoli di dover avere almeno una quantità $quant\_min_i$ di ciascuna sostanza nutritiva, cioè

$$\sum_{j \in PROD} quant\_unit_{ij} * x_j \geq quant\_min_i \quad \forall\, i \in SOST. \tag{B.2}$$

Infine di ogni prodotto dobbiamo avere una quantità non negativa, cioè

$$x_j \geq 0 \quad \forall\, j \in PROD. \tag{B.3}$$

Quindi il nostro modello matematico è il seguente problema di PL

$$\begin{array}{ll} \min & \sum_{j \in PROD} cost\_unit_j * x_j \\ & \sum_{j \in PROD} quant\_unit_{ij} * x_j \geq quant\_min_i \quad \forall\, i \in SOST \\ & x_j \geq 0 \qquad\qquad\qquad\qquad\quad \forall\, j \in PROD \end{array}$$

Una volta definito il modello matematico lo si passa ad un *risolutore* (nel nostro caso, tipicamente, l'algoritmo del simplesso discusso nel Capitolo 4) che restituisce come output (vedi Figura B.2) una soluzione ottima se questa esiste oppure segnala l'impossibilità di restutuire una soluzione ottima (nel nostro caso si possono avere le due situazioni di impossibilità per i problemi di PL: obiettivo illimitato o regione ammissibile vuota). Il passaggio dal modello matematico

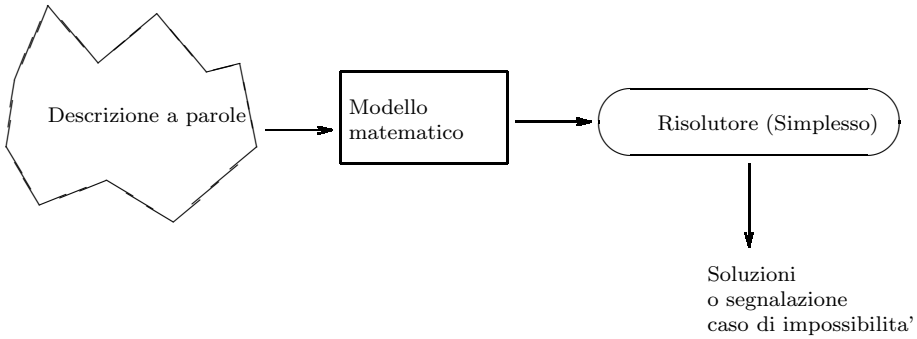


Figura B.2: Dalla descrizione a parole alla soluzione del problema.

al risolutore non è però immediato. È necessario tradurre questo modello nelle strutture dati che devono essere passate come input al programma risolutore (vedi Figura B.3). Ciò che succede nella pratica è che ci sono tipicamente grosse

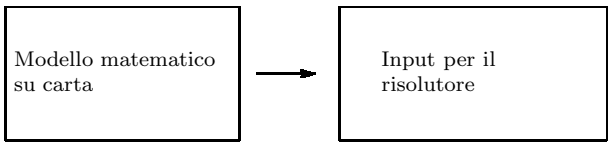


Figura B.3: Il modello matematico su carta va tradotto nell'input per il risolutore .

differenze tra il modello scritto su carta di un problema di PL e la forma in cui lo stesso modello deve essere passato come input al risolutore e la trasformazione richiede un certo sforzo. Lo scopo del linguaggio AMPL è proprio quello di facilitare questo compito. Invece di passare direttamente dal modello su carta

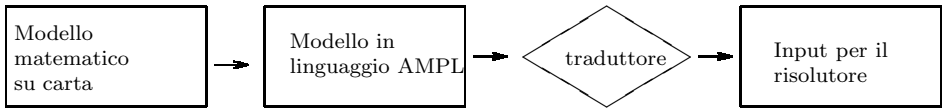


Figura B.4: Il modello in AMPL traduce il modello su carta e viene a sua volta tradotto nell'input per il risolutore.

all'input per il risolutore, si scrive un modello in linguaggio AMPL che viene poi passato a un traduttore che si occupa di trasformare il modello scritto in AMPL nell'input per il programma risolutore (vedi Figura B.4). Il vantaggio di questo passaggio supplementare è che AMPL è concepito in modo che il modello scritto in AMPL sia molto simile al modello scritto su carta. Inoltre, risolutori diversi possono richiedere input in formati molto diversi tra loro. Di tutto questo *non* si deve preoccupare l'utente AMPL, che utilizza la stessa sintassi indipendentemente dal risolutore che verrà utilizzato. L'utente si limiterà a specificare il risolutore tramite il comando

ampl: option solver *nome_risolutore*;

Sarà poi il traduttore a occuparsi della traduzione nel formato di input richiesto dal risolutore specificato. In partciolare, un risolutore largamente utilizzato per problemi di PL e PLI è *cplex*.

Verranno ora introdotte alcune parti fondamentali di AMPL. Le stesse verranno illustrate attraverso l'esempio della dieta mostrato in precedenza. In un modello di programmazione lineare (o di programmazione lineare intera) gli elementi fondamentali sono:

Insiemi di indici Nel caso del problema della dieta sono i due insiemi $PROD$

dei prodotti e $SOST$ delle sostanze nutritive.

Parametri Sono tutte le quantità il cui valore è noto prima di risolvere il problema. Nel problema della dieta sono i costi unitari $cost\_unit_j$ dei prodotti, le quantità minime $quant\_min_i$ delle sostanze nutritive e le quantità $quant\_unit_{ij}$ di sostanza i in un'unità di prodotto j.

Variabili Sono le quantità i cui valori devono essere stabiliti attraverso la soluzione del problema. Nell'esempio sono le variabili x_j indicanti le quantità di prodotto j.

Vincoli Limitano la scelta delle variabili. Nell'esempio abbiamo i vincoli (B.2) sulle quantità minime di ciascuna sostanza ed i vincoli (B.3) sulla non negatività delle quantità di ciascun prodotto.

Obiettivo È la quantità il cui valore va massimizzato (o minimizzato) scegliendo opportunamente i valori delle variabili. Nell'esempio si deve minimizzare il costo della dieta, ovvero l'obiettivo è dato da (B.1).

Si noti che questi coincidono con le componenti che abbiamo riconosciuto in un problema di decisione, con però un maggiore dettaglio: la componente dati è stata scomposta nelle due componenti insiemi di indici e parametri. Si tratta ora di vedere come questi elementi vengono dichiarati e definiti in linguaggio AMPL.

Insiemi Un insieme T si dichiara semplicemente attraverso la seguente sintassi

set T **;**

Si noti che questa è soltanto la *dichiarazione* dell'insieme. Da una qualche altra parte, come vedremo, andrà specificato il contenuto dell'insieme, ovvero verrà data la *definizione* dell'insieme.
Nel nostro esempio avremo le seguenti dichiarazioni

set $PROD$ **;**
set $SOST$ **;**

Parametri Un parametro a si dichiara nel modo seguente

param a **;**

Qualora sia noto che il parametro è sempre positivo conviene indicarlo esplicitamente nella dichiarazione nel modo seguente

param $a > 0$ **;**

In modo analogo si può specificare che un parametro è non negativo ($>= 0$), negativo (< 0) o non positivo ($<= 0$). Si può anche specificare che un parametro deve avere un valore intero nel modo seguente

param $a > 0$ **integer ;**

In questo caso il parametro a deve essere un intero positivo.
È possibile anche dichiarare vettori di parametri con indici in un insieme T attraverso la seguente dichiarazione

param $a\{T\}$ **;**

Nel caso gli indici siano gli interi da 1 a n si può scrivere

param $a\{1..n\}$ **;**

Si possono infine anche dichiarare array bidimensionali con insiemi di indici T_1 e T_2 nel modo seguente

param $a\{T_1, T_2\}$ **;**

Nell'esempio si avranno le seguenti dichiarazioni

param $cost\_unit\{PROD\} > 0$ **;**
param $quant\_unit\{SOST,PROD\} >= 0$ **;**
param $quant\_min\{SOST\} > 0$**;**

Si noti che si richiede che i costi unitari e le quantità minime siano strettamente positive, mentre le quantità di sostanza in un'unità di prodotto devono essere non negative.

Variabili La definizione delle variabili è del tutto analoga a quella dei parametri. La sola differenza è che la parola chiave **param** viene sostituita dalla parola chiave **var**. Nel nostro esempio abbiamo un vettore di variabili x con indici in $PROD$ che sarà definito in questo modo

var $x\{PROD\} >= 0$ **;**

Si noti che nella dichiarazione delle variabili sono già compresi i vincoli di non negatività (B.3) delle stesse. Nel caso una variabile sia vincolata ad assumere solo valori interi è sufficiente aggiungere la parola chiave **integer** nella sua dichiarazione, esattamente come si è fatto per i parametri. Si noti che questa aggiunta è la sola cosa che distingue un modello di AMPL per la PL da uno per la PLI. Se sulle variabili si hanno, oltre vincoli di non negatività, anche limiti superiori sui valori delle stesse, possiamo indicare tali limiti sulla stessa riga. Per esempio, se avessimo anche un parametro

param $max\_prod\{PROD\} > 0$ **;**

che definisce dei limiti superiori sui valori assunti dalle variabili, potremmo modificare la dichiarazione delle variabili come segue

var $x\{i$ **in** $PROD\} >= 0, <= max\_prod[i]$ **;**

Si noti come in questo caso abbiamo dovuto introdurre l'indice i appartenente (**in**) all'insieme $PROD$ per potersi riferire alle componenti del parametro *max_prod* (si noti anche l'uso delle parentesi quadre per richiamare tali componenti).

Vincoli Un singolo vincolo viene dichiarato nel seguente modo

subject to *nome_vincolo* **:** *formula_vincolo* **;**

Spesso però non si hanno singoli vincoli, ma collezioni di vincoli indicizzate su degli insiemi. È questo per esempio il caso dei vincoli (B.2) nel problema della dieta. Tali vincoli sono indicizzati rispetto all'insieme $SOST$. In generale, dato un insieme di indici I, la collezione di vincoli indicizzata su I viene dichiarata in questo modo

subject to *nome_insieme_vincoli* $\{i$ **in** $I\}$ **:** *formula_vincoli* **;**

Per i vincoli (B.2) si avrà la seguente dichiarazione

subject to min_sostanza $\{i$ **in** $SOST\}$ **: sum** $\{$j **in** $PROD\}$ quant_unit[i,j]*x[j] >= quant_min[i] **;**

Nella formula si deve notare l'uso di **sum** $\{j$ **in** $J\}$ per la definizione di una sommatoria con indice J.

Obiettivo L'obiettivo si dichiara, in generale, nel modo seguente

maximize *nome_obiettivo* **:** *formula_obiettivo* **;**

(nel caso si debba minimizzare si usa la parola chiave **minimize** al posto di **maximize**). Nel nostro esempio avremo

minimize total_cost **: sum** $\{j$ **in** $PROD\}$ cost_unit[j]*x[j] **;**

Abbiamo quindi definito in AMPL tutti gli elementi del nostro problema della dieta. Questi verranno trascritti in un file a cui si assegna il nome di DIETA.MOD

che si presenterà come segue (le scritte comprese tra ### sono commenti).

DIETA.MOD

INSIEMI

set $PROD$ **;**
set $SOST$ **;**

PARAMETRI

param $cost\_unit\{PROD\} > 0$ **;**
param $quant\_unit\{SOST,PROD\} >= 0$ **;**
param $quant\_min\{SOST\} > 0$**;**

VARIABILI

var $x\{PROD\} >= 0$ **;**

VINCOLI

subject to min_sostanza $\{i$ **in** $SOST\}$ **: sum** $\{j$ **in** $PROD\}$ quant_unit[i,j]*x[j] >= quant_min[i] **;**

OBIETTIVO

minimize total_cost **: sum** $\{j$ **in** $PROD\}$ cost_unit[j]*x[j] **;**

Una volta costruito il modello bisognerà inserire in un altro file i valori di insiemi e parametri, ovvero i dati di input dell'istanza del nostro problema. Mentre il modello viene inserito in un file con estensione .MOD, i valori vengono inseriti in un file con estensione .DAT. Il file DIETA.DAT dovrà contenere le definizioni degli insiemi $PROD$ e $SOST$ ed i valori assegnati ai diversi parametri. Supponiamo di avere a disposizione i seguenti dati. L'insieme $PROD$ contiene *pasta,*

verdura, carne; l'insieme $SOST$ contiene *vitamine, proteine*; un'unità di pasta, verdura e carne costano rispettivamente 3,2 e 5; le quantità minime di vitamine e proteine sono rispettivamente 8 e 6; le quantità di vitamine in un'unità di pasta, verdura o carne sono rispettivamente 0.3, 0.5 e 0.4; le quantità di proteine in un'unità di pasta, verdura o carne sono rispettivamente 0.5, 0.2 e 0.7. Bisogna ora vedere come questi dati devono essere scritti nel file DIETA.DAT. Per quanto riguarda la definizione di un insieme J contenente gli oggetti $t_1, t_2, \ldots, t_n$ si usa la seguente sintassi

set T **:=** t_1 t_2 ... t_n **;**

Nel nostro esempio avremo

set $PROD$ **:=** pasta verdura carne **;**
set $SOST$ **:=** vitamine proteine **;**

Per quanto riguarda i parametri si usa la seguente sintassi

param a **:=** *valore_parametro* **;**

Per parametri vettore con insieme indice $T = \{t_1, \ldots, t_n\}$ si usa la seguente sintassi

param a **:=**
t_1 $valore_1$
$\vdots$
t_n $valore_n$ **;**

Per parametri che sono array bidimensionali con primo insieme di indici $T = \{t_1, \ldots, t_n\}$ e secondo insieme di indici $S = \{s_1, \ldots, s_m\}$ si usa la seguente sintassi

param a **:**

$$\begin{array}{cccc} & s_1 & \cdots & s_m \quad := \\ t_1 & val(t_1, s_1) & \cdots & val(t_1, s_m) \\ \vdots & \vdots & \vdots & \vdots \\ t_n & val(t_n, s_1) & \cdots & val(t_n, s_m) \quad ; \end{array}$$

Nel nostro esempio si avranno quindi i seguenti assegnamenti.

param cost_unit **:=**
pasta 3
verdura 2
carne 5 **;**

param quant_min **:=**
vitamine 8
proteine 6 **;**

param quant_unit **:**

	pasta	verdura	carne	**:=**
vitamine	0.3	0.5	0.4	
proteine	0.5	0.2	0.7	**;**

Quindi il file DIETA.DAT per questo esempio si presenterà nella seguente forma

---

DIETA.DAT

INSIEMI

set *PROD* **:=** pasta verdura carne **;**
set *SOST* **:=** vitamine proteine **;**

PARAMETRI

param cost_unit **:=**
pasta 3
verdura 2
carne 5 **;**

param quant_min **:=**
vitamine 8
proteine 6 **;**

param quant_unit **:**

	pasta	verdura	carne	**:=**
vitamine	0.3	0.5	0.4	
proteine	0.5	0.2	0.7	**;**

Una volta inseriti i dati nel file DIETA.DAT siamo pronti per la risoluzione del problema. Prima però va fatta un'osservazione. Qui abbiamo inserito certi dati ma può capitare che lo stesso tipo di problema debba essere risolto con altri dati (ad esempio il costo di certi prodotti può cambiare o tra le sostanze se ne possono aggiungere altre come i carboidrati). AMPL è concepito in modo tale che queste modifiche possano essere fatte andando a modificare il solo file .DAT mentre nessuna modifica deve essere fatta nel file .MOD.
Ora siamo pronti per la risoluzione del problema. È sufficiente inserire i seguenti comandi in corrispondenza del prompt ampl:

```
ampl: reset;
ampl: option solver cplex;
ampl: model DIETA.MOD;
ampl: data DIETA.DAT;
ampl: solve;
```

(Si notino i ; al termine di ogni comando). Il primo comando di reset non è sempre necessario se non si sono verificati cambiamenti (o, nel caso questi riguardino solo i dati, ci si può limitare a un comando reset data;) ma, dal momento che vengono sempre tenuti in memoria l'ultimo modello e gli ultimi dati caricati, conviene usare il reset per evitare che modello e dati attuali siano sporcati da informazioni precedenti. La seconda riga, come già osservato, specifica che si richiede cplex (risolutore di problemi di PL e PLI) come risolutore. L'esecuzione di cplex resta invisibile all'utente, ma vale la pena citare il fatto che l'utente ha la possibilità di scegliere tra alcune opzioni corrispondenti a diversi modi di funzionamento per il risolutore. Se nulla viene specificato, il risolutore funziona nella modalità di default. Non ci addentreremo in questi dettagli ma rimandiamo agli appositi manuali per questo. La terza riga specifica che il modello deve essere letto dal file DIETA.MOD. La quarta riga specifica che i dati devono essere letti dal file DIETA.DAT. Infine, la quinta riga comunica ad AMPL di prendere modello e dati caricati, tradurli nell'input del risolutore e quindi risolvere il problema. A questo punto apparirà automaticamente il valore ottimo del problema (se, come in questo caso, esiste).

```
optimal solution; objective 45.263.....
```

Per visualizzare la soluzione primale si deve dare il comando

```
ampl: display x;
```

Apparirà la seguente risposta

```
x[*] :=
carne 0
pasta 7.36842
verdura 11.5789
;
```

Per visualizzare la soluzione duale (si veda il Capitolo 5), ricordando che a ogni vincolo primale corrisponde una variabile duale, è sufficiente mandare il comando display con il nome dei vincoli primali.

```
ampl: display min_sostanza;
```

Apparirà la seguente risposta

```
min_sostanza [*] :=
proteine 4.73684
vitamine 2.10526
;
```

Avremo quindi che il valore dell'ottimo è 45.2632; che nella soluzione ottima si ha una quantità 0 di carne, una quantità 7.36842 di pasta ed una quantità 11.5789 di verdura; che le variabili nella soluzione ottima del duale hanno valore 4.73684 quella relativa al vincolo sulle proteine e 2.10526 quella relativa al vincolo sulle vitamine. Si noti che, come atteso, il valore dell'ottimo duale

$$4.73684 * 6 + 2.10526 * 8 = 45.2632$$

coincide con l'ottimo del primale. Tra le altre informazioni visualizzabili ricordiamo anche i coefficienti di costo ridotto delle variabili (si veda il Capitolo 4) per visualizzare i quali basta usare il comando

```
ampl: display x.rc;
```

(l'estensione rc sta per *r*educed *c*ost), e i range nei quali non cambia la base ottima attuale per modifiche dei termini noti dei vincoli e coefficienti delle variabili nell'obiettivo (si veda il Capitolo 6 sull'analisi di sensitività). Per i termini

noti dei vincoli, il comando

ampl: display min_sostanza.down, min_sostanza.current, min_sostanza.up;

restituisce per ciascun termine noto il limite inferiore del range in cui non cambia la base ottima attuale (down), il valore attuale del termine noto (current) e il limite superiore del range (up). Analogamente, il comando

ampl: display x.down, x.current, x.up;

restituisce per ciascun coefficiente nell'obiettivo il limite inferiore del range in cui non cambia la base ottima attuale (down), il valore attuale del coefficiente (current) e il limite superiore del range (up).

Per poter calcolare tali valori è necessario attivare una option di cplex:

ampl: option cplex_options 'sensitivity';

Teniamo infine presente che, invece di digitare uno alla volta i comandi visti, possiamo anche scrivere un file DIETA.RUN in cui specifichiamo tutti questi comandi più eventualemente molti altri con costrutti tipici dei linguaggi di programmazione (statement IF, cicli, eccetera), sui quali non ci soffermeremo qui.

DIETA.RUN

```
reset;
option solver cplex;
model DIETA.MOD;
data DIETA.DAT;
solve; display x;
display min_sostanza;
display x.rc;
display min_sostanza.down, min_sostanza.current, min_sostanza.up;
display x.down, x.current, x.up;
```

Una volta scritto il file DIETA.RUN possiamo eseguire tutti i comandi in esso contenuti semplicemente con il comando

ampl: include DIETA.RUN;

B.1 Un esempio di PLI

Vogliamo ora introdurre un altro esempio di traduzione di un modello su carta in un modello in linguaggio AMPL. In questo caso si prenderà in considerazione un problema di programmazione lineare intera, il problema dello zaino. Come già fatto per il problema della dieta, si darà dapprima una descrizione a parole del problema, quindi lo si tradurrà in un modello di PLI su carta ed infine si passerà alla traduzione in linguaggio AMPL.

Descrizione a parole È dato un insieme $OGGETTI$ di oggetti a ciascuno dei quali è associato un *peso* e un *valore*. È inoltre dato uno zaino a cui è associata una *capacità*, ovvero un peso massimo che può essere trasportato all'interno dello zaino. Si vogliono inserire degli oggetti nello zaino in modo tale da massimizare il valore complessivo trasportato in esso, tenendo conto che il peso totale degli oggetti inseriti non può superare la capacità dello zaino.

Modello su carta All'oggetto i associamo una variabile binaria x_i, ovvero una variabile che può assumere solamente i valori 0 e 1. Se $x_i = 0$ l'oggetto i non viene inserito nello zaino, se $x_i = 1$ l'oggetto viene inserito nello zaino. L'unico vincolo è quello che gli oggetti inseriti nello zaino abbiano un peso complessivo che non superi la capacità dello zaino, ovvero

$$\sum_{i \in OGGETTI} peso_i * x_i \leq capacità.$$

L'obiettivo è quello di massimizzare il valore degli oggetti inseriti nello zaino, quindi

$$\max \sum_{i \in OGGETTI} valore_i * x_i.$$

Il modello su carta avrà dunque la seguente forma

$$\begin{array}{ll} \max & \sum_{i \in OGGETTI} valore_i * x_i \\ & \sum_{i \in OGGETTI} peso_i * x_i \leq capacità \\ & x_i \in \{0,1\} \qquad\qquad i \in OGGETTI \end{array}$$

Modello in linguaggio AMPL abbiamo un solo insieme, l'insieme $OGGETTI$, che verrà dichiarato nel modo seguente

set $OGGETTI$ **;**

Come parametri abbiamo due vettori di parametri, *peso* e *valore*, con insieme indice $OGGETTI$ ed il singolo parametro *capacità*. Tutti questi parametri sono positivi e verranno dichiarati nel modo seguente

param $peso\{OGGETTI\} > 0$ **;**
param $valore\{OGGETTI\} > 0$ **;**
param $capacità > 0$ **;**

Avremo poi un vettore di variabili con insieme indice $OGGETTI$. Le variabili sono vincolate ad assumere i soli valori 0 e 1. Si può esprimere questo con la seguente dichiarazione delle variabili

var $x\{OGGETTI\} >= 0, <= 1,$ **integer ;**

In questo modo le variabili sono vincolate ad essere interi compresi tra 0 e 1 e quindi possono assumere i soli valori 0 e 1. Tuttavia, visto il largo uso che si fa nella PLI di variabili binarie, AMPL prevede una dichiarazione speciale per esse:

var $x\{OGGETTI\}$ **binary ;**

Per quanto riguarda i vincoli, nel problema dello zaino abbiamo solo quello sulla capacità dello zaino che verrà dichiarato nel modo seguente

subject to max_capac **:** $\text{sum}\{i$ **in** $OGGETTI\}\ peso[i]*x[i] <= capacità$ **;**

Infine l'obiettivo verrà dichiarato nel modo seguente

maximize tot_valore **:** $\text{sum}\{i$ **in** $OGGETTI\}\ valore[i] * x[i]$ **;**

Mettendo assieme quanto visto, il file contenente il modello, che chiameremo ZAINO.MOD, si presenterà nella seguente forma

---

ZAINO.MOD

INSIEMI

set $OGGETTI$ **;**

PARAMETRI

param $peso\{OGGETTI\} > 0$ **;**
param $valore\{OGGETTI\} > 0$ **;**
param $capacità > 0$ **;**

VARIABILI

var $x\{OGGETTI\}$ **binary ;**

VINCOLI

subject to max_capac **:** $\text{sum}\{i \textbf{ in } OGGETTI\}\ peso[i] * x[i] <= capacità$ **;**

OBIETTIVO

maximize tot_valore **:** $\text{sum}\{i \textbf{ in } OGGETTI\}\ valore[i] * x[i]$ **;**

Una volta costruito il modello si tratta di considerare delle particolari istanze di problema dello zaino in cui siano date le definizioni dell'insieme $OGGETTI$ e dei parametri. Ad esempio, consideriamo un caso in cui gli oggetti sono un *libro*, una *radio*, un *telefono* e una *macchina fotografica*, con peso rispettivamente pari a 3, 7, 2 e 4 e valore 5, 8, 4 e 7. La capacità dello zaino è 11. Il file di dati ZAINO.DAT si presenterà come segue

---

ZAINO.DAT

INSIEMI

set $OGGETTI$ **:=** libro radio telefono macchina_fotografica **;**

PARAMETRI

param peso **:=**
libro 3
radio 7
telefono 2
macchina_fotografica 4 **;**

param valore **:=**
libro 5
radio 8
telefono 4
macchina_fotografica 7 **;**

param capacità := 11 **;**

A questo punto per risolvere il problema si procede come già visto per il problema della dieta, inserendo modello e dati e chiamndo il risolutore attraverso i seguenti comandi

```
ampl: reset;
ampl: option solver cplex;
ampl: model ZAINO.MOD;
ampl: data ZAINO.DAT;
ampl: solve;
```

Il programma risolutore è cplex. Anche qui ricordiamo che, oltre al funzionamento di default del risolutore, l'utente ha la possibilità di scegliere tra alcune opzioni corrispondenti a diversi modi di funzionamento. Attraverso il comando

```
ampl: display x;
```

è possibile visualizzare la soluzione ottima. Nell'esempio specifico essa è

x[*] :=

```
libro 1
macchina_fotografica 1
radio 0
telefono 1
;
```

con valore ottimo tot_valore pari a 16. Quindi la soluzione ottima ha valore 16 e consiste nel mettere nello zaino il libro, la macchina fotografica ed il telefono mentre la radio viene lasciata fuori. Anche in questo caso potremmo raccogliere tutti i comandi in un appposito file ZAINO.RUN.

Nota

Va ribadito che questi appunti contengono solo alcune idee di base di AMPL ma il linguaggio è molto più ricco e consente costruzioni anche molto più complicate di quelle viste qui. Alcune possibilità fornite dal linguaggio verranno esplorate nel Capitolo 2 sui modelli. Per tutto quello che non si trova in questi appunti si rimanda al manuale *AMPL: a Modeling Language for Mathematical Programming*.